GHI forecasting to extend the range of a solar powered vehicle.

by

Carel Landman

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electrical Engineering) in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr. A.J. Rix

March 2020

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2020

Abstract

GHI forecasting to extend the range of a solar powered vehicle.

C. Landman

Department of Electrical and Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MEng (ElectricalEngineering)

March 2020

The focus of this project is to investigate, model and optimise the range of electric vehicles(EVs). To do so, the detailed modelling of an EV (2016 Nissan Leaf) is covered, to simulate the range of the EV and to investigate how different factors will influence the range. Further, research was done to investigate whether mounting photovoltaic (PV) panels on an EV will extend the range, and if so, by how much. To accurately review the contribution from PV panels mounted on an EV, global horizontal irradiance (GHI) forecasting is done, by only making use of data that is freely available to the public. The GHI forecasting is based on the relationship between clearsky GHI, actual GHI and cloud cover and is done by making use of regression techniques.

The biggest factors effecting the range of an EV is the incline of the road and the aerodynamic drag of the vehicle. Since 2016, batteries developed so much, that a battery with double the energy density of the battery in the 2016 Nissan Leaf, is now available. If this battery is used, the range of the EV will effectively be doubled. The GHI forecasting was compared to an existing model and shows more accurate forecasts than this model. It was concluded that PV panels mounted on an EV will lead to a very small increase in range as the EV drives, where the battery-charge gained as the EV is parked, has a more positive effect.

Uittreksel

GHB voorspelling om die reikafstand van 'n son-aangedrewe voertuig te verleng.

("GHI forecasting to extend the range of a solar powered vehicle.")

C. Landman

Departement Elektries en Elektroniese Ingenieurswese,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MEng (ElektrieseIngenieurswese)

Maart 2020

Die fokus van hierdie projek is om die reikafstand van elektriese voertuie (EV) te ondersoek, te modelleer en te optimaliseer. Om dit te kan doen, word die gedetailleerde modellering van 'n EV (2016 Nissan Leaf) gedek, om die reikafstand van die EV te simuleer en om te ondersoek hoe verskillende faktore die reikafstand sal beïnvloed. Verder is navorsing gedoen om te ondersoek of die montering van fotovoltaïese (FV) panele op 'n EV die reikafstand sal vergroot, en indien wel, met hoeveel. Om die bydrae van FV-panele wat op 'n EV gemonteer is, akkuraat na te vors, word voorspellings gedoen vir globale horisontale bestralings (GHB), deur slegs gebruik te maak van data wat vrylik vir die publiek beskikbaar is. Die GHB-voorspelling is gebaseer op die verwantskap tussen skoon-dag GHB, werklike GHB en wolkbedekking en word gedoen deur gebruik te maak van regressietegnieke.

Die grootste faktore wat die omvang van 'n EV beïnvloed, is die helling van die pad en die aërodinamiese weerstand van die voertuig. Sedert 2016 het batterye soveel gevorder, dat daar nou 'n battery met dubbel die energiedigtheid van die battery in die 2016 Nissan Leaf is. As hierdie battery gebruik word, sal die reikafstand van die EV effektief verdubbel word. Die GHB-voorspelling is vergelyk met 'n bestaande model en toon meer akkurate voorspellings as hierdie model. Daar is tot die gevolgtrekking gekom dat PV-panele wat op 'n EV gemonteer is, tot 'n baie klein toename in reikafstand sal lei, terwyl die EV ry, maar die battery-lading wat verkry word as die EV geparkeer is, 'n meer positiewe uitwerking het.

Acknowledgements

I would like to express my sincere gratitude to the following parties:

- My supervisor, DR A.J. Rix, for his guidance and encouragement during this project.
- My parents, for providing financial support for me in order to complete my postgraduate studies.
- My family, for their continuous support throughout the whole project.
- My friends, for giving advice and support along the way.
- The MIH Media lab, for providing a productive working environment.

List of Publications

- Landman, C. and Rix, A.J.: Performance Prediction for an Electric Vehicle. In: *2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, Bloemfontein, South Africa, 2019, pp. 538−543.
- Landman, C. and Rix, A.J.: Using cloud cover forecasts for estimating a solar powered vehicles′ range. In: *2020 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, Cape Town.[Accepted, awaiting publication]

Dedications

Hierdie tesis word opgedra aan my ouers, Andries Erasmus en Saritha Landman.

Contents

List of Figures

List of Tables

Nomenclature

Constants

ρ_{air}	Air Density (1.255)	$[\,\mathrm{kg \cdot m^{-3}}\,]$
k	Boltzmann Constant (1.3807 x 10^{-23})	$[\,\mathrm{J \cdot K^{-1}}\,]$
q	Electron Charge (1.602 x 10^{-19})	$[\,\mathrm{C}\,]$
g	Gravitational Acceleration (9.81)	$[\,\mathrm{m \cdot s^{-2}}\,]$
G_n	Irradiance at STC (1000)	$[\,\mathrm{W/m^2}\,]$
SC	Solar Constant (1364)	$[\,\mathrm{W/m^2}\,]$
C	Speed of light (299 792 458)	$[\,\mathrm{m/s}\,]$
T_n	Temperature at STC (25)	$[\,\mathrm{°C}\,]$

Variables

F_{ad}	Aerodynamic Drag	$[\,\mathrm{N}\,]$
m_a	Air Mass	$[\,-\,]$
D	Altitude Dependant Coefficient	$[\,-\,]$
Alt	Altitude or Elevation	$[\,\mathrm{m}\,]$
Alt	Altitude	$[\,\mathrm{m}\,]$
T_{amb}	Ambient Air Temperature	$[\,\mathrm{°C}\,]$
θ_{AOI}	Angle of Incidence	$[\,\mathrm{°}\,]$
$F_{\omega a}$	Angular Acceleration Force	$[\,\mathrm{N}\,]$
ω_{EV}	Angular Velocity of Wheels	$[\,\mathrm{m \cdot s^{-1}}\,]$
W	Atmospheric Precipitable Water	$[\,\mathrm{cm}\,]$
$\tilde{x}$	Average of X in Dataset	$[\,-\,]$
$\tilde{y}$	Average of Y in Dataset	$[\,-\,]$
B_{cap}	Battery Capacity	$[\,\mathrm{kWh}\,]$
I_{bat}	Battery Current	$[\,\mathrm{A}\,]$
R_{int}	Battery Internal Resistance	$[\,\Omega\,]$
E_{bat}	Battery Internal Voltage	$[\,\mathrm{V}\,]$
V_{bat}	Battery Terminal Voltage	$[\,\mathrm{V}\,]$
A_{cable}	Cable Cross-sectional Area	$[\,\mathrm{m^2}\,]$

L_{cable}	Cable Length	[m]
R_{DC}	Cable Resistance	[Ω]
ΔAlt	Change in Elevation	[m]
Δv	Change in EV speed	[m/s]
C_C	Charge Capacity	[kWh]
GHI_{clear}	Clear Sky GHI	[W/m^2]
Kt^1	Clearness Index	[−]
μ_{rr}	Coefficient of Rolling Resistance	[−]
ρ_{cable}	Conductor Resistivity	[Ωm]
B_{Head}	Constant	[−]
CE	Coulombic Efficiency	[%]
P_{DC}	DC Cable Loss	[W]
δ	Declination Angle	[°]
$\bar{Y}$	Dependant Variable Matrix	[−]
Y	Dependant Variable	[−]
T_d	Dew-Point Temperatures	[°C]
α	Diode Ideality Factor	[−]
I_o	Diode Saturation Current	[A]
C_D	Discharge Capacity	[kWh]
$Dist$	Distance Between Points	[m]
C_d	Drag Coefficient	[−]
R_{Earth}	Earth Mean Radius	[m]
ΔT_{emp}	Empirically Calculated Coefficient	[°C]
E	Energy	[J]
a_{EV}	EV Acceleration	[m · s^{-2}]
v_i	EV Initial Speed	[m/s]
G_{EV}	Gear Ratio	[−]
n_{EV}	Gear System Efficiency	[−]
Kt	Global Horizontal Transmittance	[−]
$Head$	Heading	[°]
F_{hc}	Hill Climbing Force	[N]
H	Hour Angle	[°]
$\bar{X}$	Independent Variable(s) Matrix	[−]
X	Independent Variable(s)	[−]
β_0	Intercept	[−]
E_a	Internal Voltage	[V]

G	Irradiance	$[\mathrm{W/m^2}]$
$L_{1,2}$	Latitude Points	$[^\circ\mathrm{N}]$
L	Latitude	$[^\circ\mathrm{N}]$
l	Likelihood	$[-]$
F_{la}	Linear Acceleration Force	$[\mathrm{N}]$
T_L	Linke Turbidy Coefficient	$[-]$
$Long_{1,2}$	Longitude Points	$[^\circ\mathrm{E}]$
ψ_m	Magnetic Flux	$[\mathrm{Wb}]$
X_m	Magnetizing Reactance	$[\Omega]$
m_{EV}	Mass of The EV	$[\mathrm{kg}]$
m	Mass	$[\mathrm{kg}]$
T_{mod}	Module Operating Temperature	$[^\circ\mathrm{C}]$
I_{EV}	Moment of Inertia of Motor	$[\mathrm{kg \cdot m^2}]$
N_s	Number of cells connected in series	$[-]$
n	Number of Current Day	$[-]$
V_{ocn}	Open Circuit Voltage at STC	$[\mathrm{V}]$
T_{cell}	Operating temperature of the solar cell	$[^\circ\mathrm{C}]$
ρ_m	Optical Air Mass	$[-]$
R_p	Parallel- or Shunt Resistance	$[\Omega]$
θ_{load}	Phase Angle	$[^\circ]$
I_{pn}	Photocurrent at Standard Test Conditions(STC)	$[\mathrm{A}]$
I_p	Photocurrent	$[\mathrm{A}]$
P_{EV}	Power of EV	$[\mathrm{W}]$
P_{PV}	Power Produced by PV Panels	$[\mathrm{W}]$
p	Probability	$[-]$
R_2	Referred Rotor Resistance	$[\Omega]$
$\bar{\beta}$	Regression Coefficient Matrix	$[-]$
β	Regression Coefficients	$[-]$
θ_{Road}	Road Angle	$[\mathrm{rad}]$
F_{rr}	Rolling Resistance Force	$[\mathrm{N}]$
R_s	Series Resistance	$[\Omega]$
I_{scn}	Short Circuit Current at STC	$[\mathrm{A}]$
K_i	Short circuit temperature coefficient	$[\%/^\circ\mathrm{C}]$
s	Slip	$[-]$
β_1	Slope	$[-]$
ψ	Solar Azimuth Angle	$[^\circ]$

β_t	Solar Panel Tilt Angle	$[^\circ]$
θ_z	Solar Zenith Angle	$[^\circ]$
ΔKt^1	Stability Index	$[-]$
SOC	State Of Charge	$[\%]$
R_1	Stator Effective Resistance	$[\Omega]$
ρ	Surface Albedo	$[-]$
ψ_s	Surface Azimuth Angle	$[^\circ]$
X_s	Synchronous Reactance	$[\Omega]$
ω_{sync}	Synchronous Speed	$[\mathrm{m} \cdot \mathrm{s}^{-1}]$
V_a	Terminal Voltage	$[\mathrm{V}]$
Z_{TH}	Thevenin Equivalent Impedance	$[\Omega]$
X_{TH}	Thevenin Equivalent Reactance	$[\Omega]$
R_{TH}	Thevenin Equivalent Resistance	$[\Omega]$
V_{TH}	Thevenin Equivalent Voltage	$[\mathrm{V}]$
T_{PV}	Time	$[\mathrm{s}]$
T_{EV}	Torque of EV	$[\mathrm{Nm}]$
F_{te}	Tractive Effort	$[\mathrm{N}]$
A_{EV}	Vehicle Front Area	$[\mathrm{m}^2]$
v	Vehicle Speed	$[\mathrm{m} \cdot \mathrm{s}^{-1}]$
r	Wheel Radius	$[\mathrm{m}]$
WS	Windspeed	$[\mathrm{m/s}]$

Abbreviations

AC	Alternating Current	$[-]$
API	Application Programming Interface	$[-]$
CC	Cloud Cover	$[-]$
DoD	Depth of Discharge	$[\%]$
DHI	Diffuse Horizontal Irradiance	$[\mathrm{W/m}^2]$
DNI	Direct Normal Irradiance	$[\mathrm{W/m}^2]$
EV	Electric Vehicle	$[-]$
ET	Equation of Time	$[-]$
ETD	Estimated Time of Arrival	$[\mathrm{hh:mm}]$
ER	Extraterrestrial Radiation	$[\mathrm{W/m}^2]$
GHI	Global Horizontal Irradiance	$[\mathrm{W/m}^2]$
GPS	Global Positioniong System	$[-]$
GUI	Graphical User Inteface	$[-]$
GHG	Greenhouse Gasses	$[-]$

ICE	Internal Combustion Engine	[−]
KPI	Key Performance Indicator	[−]
LST	Local Solar Time	[hours]
$LSMT$	Local Standard Time Meridian	[hours]
MAE	Mean Absolute Error	[−]
$MAPE$	Mean Absolute Percentage Error	[−]
MBE	Mean Bias Error	[−]
PV	Photovoltaic	[−]
$RMSE$	Root Mean Squared Error	[−]
TC	Time Correction Factor	[−]
$POA_{diffuse}$	Total In-plane Diffuse Component	[W/m^2]
POA_{direct}	Total In-plane Direct Component	[W/m^2]
POA	Total In-plane Irradiance	[W/m^2]
UTC	Universal Coordinate Time	[hours]

Chapter 1

Introduction

1.1 Project Background

During 2010, the transport sector was consuming 32% of the total energy consumption of South Africa. The National Transport Master Plan 2050 estimates that this number will rise drastically, and that by 2050 the transport sector will be using 44% of the total energy yield [1]. This raises a big concern, as the transport sector will soon be using the majority of the annual energy yield of South Africa. As the energy demand of the transport sector rises, the amount of greenhouse gasses (GHG) emissions will rise, as internal combustion engines (ICEs) rely on the combustion of fuel for the production of energy. A solution is thus needed in order to decrease GHG emissions, without having a negative affect on the economic growth of the country.[2]

Electrical vehicles is a viable solution to this problem, as they have significantly less GHG emissions compared to ICEs (this does not include the emissions with regards to the electricity generation for the charging of the batteries), as they do not rely on the combustion of fuel. The CSIR conducted a study in 2017 that investigates the impact of an EV fleet on the annual GHG emissions in South Africa. Three different scenarios for the adoption of EVs was investigated, namely: Low adoption that refers to less than 2.8 million EVs by 2050; moderate adoption that refers to 2.8 million EVs by 2050 and high adoption that refers to 5 million EVs by 2050. It was concluded that the accumulated GHG-emissions mitigated, can be as high as 197.82 million tons by the year 2050 [1]. Figure 1.1 shows the mitigated GHG emissions for three different scenarios of EV fleet adoption.[2]

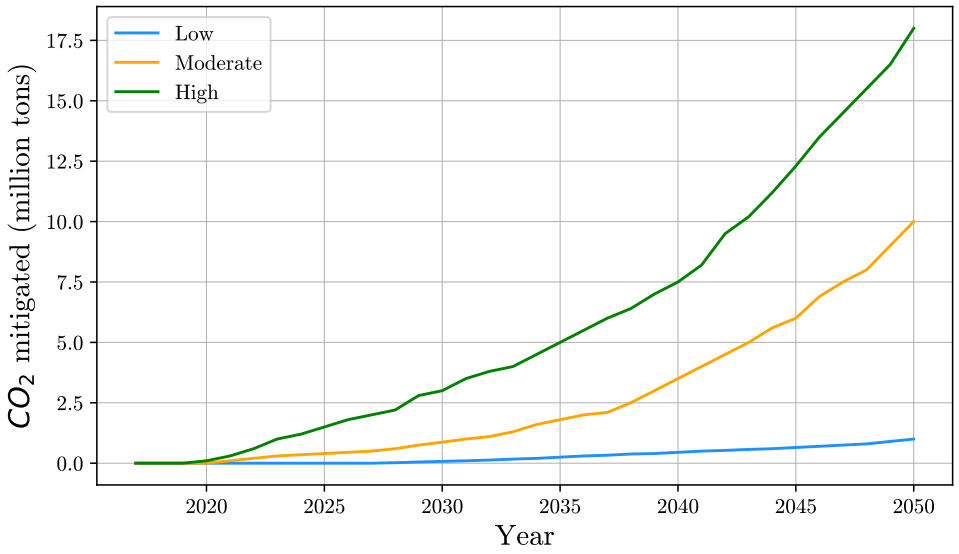


Figure 1.1: Annual Mitigated GHG Emissions [1].

There are two factors that currently hinders the widespread adoption of EVs, and they are range anxiety and the price of EVs. EVs are currently quite expensive, as the price of batteries are very high. It is generally accepted that with the development in technology regarding a product, the price will decrease as supply and demand comes into play. Table 1.1 shows a few of the currently available EVs and their corresponding ranges [1]. It is evident that range anxiety is a real concern, as the range of the majority of the EVs is below 200 km.[2]

Table 1.1: Ranges of various EVs

Vehicle	Range (km)
BMW Active E	151
BMW i3 2014-2016	135
BMW i3 2017	130
Chevy Bolt	383
Chevy Spark	132
Coda	201
Fiat 500E	140
Ford Focus Electric Vehicle	122
Kia Soul	150
Mercedes B Class B250e	140
Mitsubishi i-MiEV	100
Nissan Leaf (3.3 kW)	135
Nissan Leaf (6.6 kW)	135
2017 Nissan Leaf (3.3 kW, S Model)	172
2017 Nissan Leaf (6.6 kW, SL & SV Model)	172
Smart Car	109
Tesla Model S 60 Single	338
Tesla Model S 70 Single	386
Tesla Model S 85 Single	426
Tesla Model S 90 Single	473
Tesla Model S 100 Single	613
Tesla Model S 70 Dual	440
Tesla Model S 85 Dual	426
Tesla Model S 90 Dual	531
Tesla Model S 100 Dual	507
Tesla Model X 100 Standard	475
Toyota Rav 4	165
VW e-Golf Upgrade (3.6 kW)	133
VW e-Golf Upgrade (7.2 kW)	133

1.2 Problem Statement

Range anxiety can be described as the fear that a driver of an EV experiences when the driver is concerned that the batteries will be completely drained before the destination is reached [3]. The focus of this project is to reduce the range anxiety that the driver of an EV experiences.

1.3 Research Goal

The validity of the effect of PV panels mounted on the EV, on the range of EVs is not very concrete. This project aims to investigate whether mounting 2 PV panels on an EV is a viable solution to extend the range thereof.

1.4 Research Objectives

- Detailed modelling of a 2016 Nissan Leaf and to investigate the effect of different factors on the range thereof.
- Detailed modelling of a PV panel, including the solar angles in order to accurately calculate the total in-plane irradiance.
- GHI forecasting based on cloud cover, using freely available data.
- Developing algorithms to predict the range of an EV, to give the driver optimal driving speeds and to determine the contributions of the PV panels.
- Developing a graphical user interface (GUI), as a way of interactively showing how the different parts of the project is integrated together.

1.5 Thesis Layout

The layout of the remainder of the thesis is as follows:

1.5.1 Chapter 2

This chapter will cover the development of an accurate model of an EV, in this case a 2016 Nissan Leaf. This includes a brief overview of several battery topologies, an overview of electric motors and the modelling thereof. The modelling of the battery and the electric motor will be approached very simplistic, as the scope of the project is not on the detailed modelling of batteries and electric motors, but rather on the EV as a whole. The chapter will conclude with a sensitivity analysis on the range of the EV.

1.5.2 Chapter 3

A deep investigation on solar power and the development of an accurate model to predict the power yield of the chosen PV panel, is discussed. This will include an investigation on the different irradiance types as well as the solar angles, in order to accurately predict the total in-plane irradiance. The chapter concludes with an investigate on the losses regarding PV power generation and a sensitivity analysis on the power yield of a PV panel.

1.5.3 Chapter 4

GHI forecasting based on cloud cover and the theoretical clear-sky model, is discussed in this chapter. The forecasting will be done by implementing regression techniques and the data that will be used has to be freely available to the public. For reference, the forecasting techniques will be compared to a existing GHI forecasting model.

1.5.4 Chapter 5

The objective of this chapter is to develop software that will tie all of the individual aspects of the project together into a working system. This will include the following:

- Developing road profile data that will include all of the necessary information to model the range of the EV as well as the yield of the PV panels,
- Developing an algorithm that will give the user optimal driving speeds that will correspond with a specific range for a specific route,
- Developing an algorithm to determine the energy requirements in order to complete a specific route with certain limitations defined and
- Developing algorithms to calculate the contribution of the PV panels on the range of the EV while driving, as well as being parked.

1.5.5 Chapter 6

This chapter will discuss the results obtained through the project. The development of a graphical user interface will also be discussed.

1.5.6 Chapter 7

Project conclusions and recommendations for further work regarding the topic, is discussed.

Chapter 2

Electric Vehicles

2.1 Introduction

Electric vehicles have been around as early as the 1830s. Back when EVs were first created, rechargeable batteries were not invented yet. Near the end of the 19th century, rechargeable batteries were invented and became more widely available, this is when EVs gained popularity [4]. During the 20th century, EVs lost popularity to ICEs, as the range thereof was superior. Another disadvantage of EVs were that batteries were very expensive and this was undesirable. During the end of the 20th century, EVs regained popularity as people became more aware of environmental issues. Along with this, the technology regarding batteries was on the up-rise and they became smaller and cheaper [4].

This chapter will cover the full modelling of a 2016 Nissan Leaf, which includes the power requirements to drive along a certain route, the electric motor- and battery modelling. The chosen EV was picked, as the specification regarding the EV is accurate and available to the public. The chapter will conclude with a sensitivity analyses which aims to show how different factors will effect the predicted range of the EV.

2.2 EV Modelling

The goal for the modelling of the EV, is to create a dynamic mathematical model that will include all of the parameters that will affect the range of the modelled EV. This mathematical model is set up in such a way, that the different parameters can easily be changed. In turn, the effect thereof on the range of the EV can be investigated hassle free [4]. The EV modelling is a very important part of this project, as an inaccurate model will lead to inaccurate predictions for everything dependant on the EV model. The specifications of the 2016 Nissan Leaf can be seen in Table 2.1.

Table 2.1: 2016 Nissan Leaf Specifications [5], [6]

Description	Value
Coefficient of Rolling Resistance	0.0075
Mass of EV	1521 kg
Front Area of Vehicle	2.744 m^2
Drag Coefficient	0.28
Gear Ratio	7.9377
Wheel Radius	0.216 m
Motor Maximum Power	80 kW
Battery Capacity	30 kWh
Battery Weight	294 kg

Before the EV can be modelled, it is important to understand what forces acts on a vehicle that moves up an incline, as can be seen in Figure 2.1 [7]. The force moving the vehicle forward, is known as the tractive force or the tractive effort. For the vehicle to move forward, the tractive force has to be larger than the sum of all the forces acting against it. The tractive force can be expressed by Equation 2.1.[4]

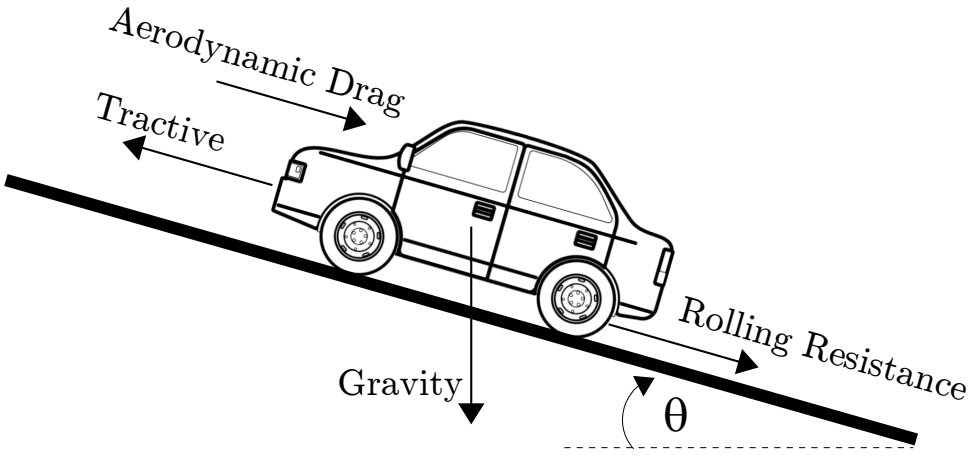


Figure 2.1: Forces Acting On a Moving Vehicle. (Adapted from [7], [8])

$$F_{te} = F_{rr} + F_{ad} + F_{hc} + F_{la} + F_{\omega a} \tag{2.1}$$

- F_{te} - Tractive Effort (N).
- F_{rr} - Rolling Resistance Force (N).
- F_{ad} - Aerodynamic Drag (N).
- F_{hc} - Hill Climbing Force (N).
- F_{la} - Linear Acceleration Force (N).
- $F_{\omega a}$ - Angular Acceleration Force (N).

2.2.1 Rolling Resistance Force

Rolling resistance force is seen as the friction between the tyres of the EV and the road. This force acts in the opposite direction that the EV is intended to- or is moving in [4]. The rolling resistance force can be calculated by Equation 2.2 [6]. As can be derived by Equation 2.2, the two main factors that can influence the rolling resistance force is the mass of the vehicle and the coefficient of rolling resistance, as the gravitational acceleration stays constant. The total mass of the EV can be controlled to a certain extent, in the way that the driver can control the number of passengers and/or luggage.

The coefficient of friction is dependant on the tyre type, size and pressure. For this project, the coefficient of friction is seen as constant. The coefficient of friction of a tyre can be as high as 0.015 for general tyres and as low as 0.005 for low-friction tyres developed especially for EVs [4].

$$F_{rr} = \mu_{rr} \cdot m \cdot g \tag{2.2}$$

- μ_{rr} - Coefficient of Rolling Resistance.
- m_{EV} - Mass of The Vehicle (kg).
- g - Gravitational Acceleration ($m \cdot s^{-2}$).

2.2.2 Aerodynamic Drag

The friction between the body of a vehicle with the air it is moving through, is defined as aerodynamic drag. This force acts in the opposite direction that the EV is moving in [4]. The aerodynamic drag can be determined by Equation 2.3. The drag coefficient is generally determined by placing the vehicle in a wind tunnel and running tests thereon. The drag coefficient is dependant on the shape of the vehicle, protrusions and the front area of the vehicle. The value for the air density is chosen as 1.255 $kg \cdot m^{-3}$, as this is considered as the air density at standard temperature and pressure.[9]

$$F_{ad} = \frac{1}{2} \cdot \rho_{air} \cdot A_{EV} \cdot C_d \cdot \upsilon^2 \tag{2.3}$$

- ρ_{air} - Air Density $(kg \cdot m^{-3})$.
- C_d - Drag Coefficient.
- A_{EV} - Vehicle Front Area (m^2).
- υ - Vehicle Speed $(m \cdot s^{-1})$.

2.2.3 Hill Climbing Force

The force that is caused by the weight of the EV acting along the slope, is known as the hill climbing force [4]. The hill climbing force is in some cases the biggest force to overcome in order for the vehicle to start moving. The hill climbing force can be determined by Equation 2.4, where α_{road} is the road angle in radians [9].

$$F_{hc} = m \cdot g \cdot \sin(\alpha_{road}) \tag{2.4}$$

2.2.4 Linear Acceleration Force

The linear acceleration force is derived from Newton's second law, which states: "The rate of change of momentum of a body is proportional to the resultant force acting on it and takes place in the direction of that force"[10]. The linear acceleration force, is the force needed to accelerate or decelerate a vehicle. The linear acceleration force can be calculated by Equation 2.5, where a_{EV} is the acceleration of the vehicle [9].

$$F_{la} = m \cdot a_{EV} \tag{2.5}$$

2.2.5 Angular Acceleration Force

There is a force needed to increase the acceleration of the rotational axis of the EVs electric motor, and this force is called the angular acceleration force. The angular acceleration force can be determined by Equation 2.6 [6]. As in most cases, a reliable source for the moment of inertia of the motor and the gear efficiency cannot be found for the modelled EV, thus the angular acceleration force was not taken into consideration. The angular acceleration force is quite small compared to the linear acceleration force, only about 5% thereof [4]. Thus it is deemed acceptable to neglect the angular acceleration force in this project.

$$F_{\omega a} = \frac{I_{EV} \cdot G_{EV}^2}{n_{EV} \cdot r^2} \cdot a_{EV} \tag{2.6}$$

- I_{EV} - Motors' Moment of Inertia ($kg \cdot m^2$).
- G_{EV} - Gear Ratio.
- n_{EV} - Gear System Efficiency.
- r - Wheel Radius (m).

2.2.6 Power Requirements

To determine the power requirements of a specific route, the relationship between power and tractive effort needs to be understood. The total tractive effort as determined by Equation 2.1, can also be expressed by Equation 2.7 [4]. From Equation 2.7, the torque requirements can now be determined, but the power requirements still needs to be calculated. This can be done by implementing Equation 2.8 [6].

$$F_{te} = \frac{G_{EV} \cdot T_{EV}}{r} \tag{2.7}$$

$$P_{EV} = \omega_{EV} \cdot T_{EV} \tag{2.8}$$

- T_{EV} - Torque (Nm).
- P_{EV} - Power (W).
- ω_{EV} - Angular Velocity of Wheels ($m \cdot s^{-1}$).

2.3 Electric Motors

As the name, Electric Vehicle suggests, an EV gets its power from an electric motor. An electric motor converts electrical energy into mechanical energy, where as an internal combustion engine relies on the burning of fuel to create mechanical energy. The chosen EV makes use of a 80 kW synchronous motor. The synchronous motor as well as an induction motor will be discussed in short. The induction motor is discussed for comparison purposes.

Put very simply, an electric motor consists of a moving and stationary component, namely the rotor and the stator. The main difference between an induction- and a synchronous motor is that: In a synchronous motor the rotor flux is produced by DC-excitation of the field windings(DC-excited motor) or by permanent magnets(Non-excited motor). In an induction motor, rotor currents are induced within the rotor windings by the time-varying stator currents and the rotors' motion relative to the stator, which in turn produces a flux within the rotor [11].

2.3.1 Synchronous Motor

In synchronous motors, the flux generated within the rotor produces a pair of magnetic poles on the rotor. The alternating current (AC) connected to the windings of the stator produces a rotating magnetic field in the stator windings which interacts with the magnetic poles of the rotor. This interaction causes the rotor to rotate [12]. The equivalent circuit of a synchronous motor can be seen in Figure 2.2.

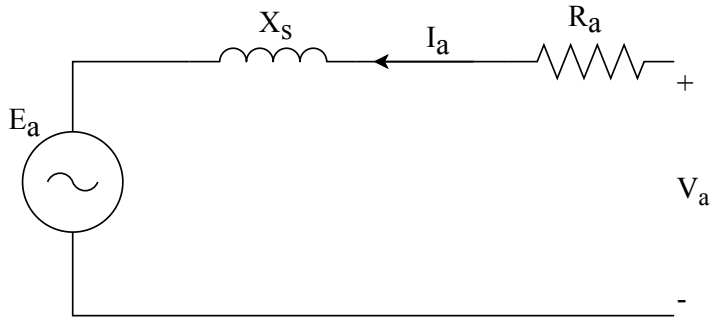


Figure 2.2: Synchronous Motor Equivalent Circuit. (Adapted from [11])

The torque developed by a synchronous motor can be determined by understanding the equivalent circuit and making use of Equation 2.9 [11]. Figure 2.3 shows the power and torque vs speed of the electric motor of the chosen EV. From this figure, it can be seen that a synchronous motor produces constant torque up until synchronous speed, from where it then produces constant power. At synchronous speed, the maximum terminal voltage is applied, hence the rotor cannot rotate faster. The motor speed can be determined by Equation 2.10, where ψ_m is the magnetic flux between the stator and rotor. From Equation 2.10, it can be derived that the only way to increase the speed beyond the synchronous speed, is to decrease this magnetic flux. This is known as field-weakening. Field-weakening is done by applying an opposing magnetic field in the stator coils, which will reduce the magnetic field between the stator and the rotor, and the motor will be able to rotate faster than synchronous speed.[13]

$$T_{EV} = \frac{V_a \cdot E_a}{X_s \cdot \omega_{EV}} \cdot \sin(\theta_{load}) \tag{2.9}$$

- V_a - Terminal Voltage (V).
- E_a - Internal Generated Voltage (V).
- X_s - Synchronous Reactance (Ω).
- θ_{load} - Phase Angle ($^\circ$).

$$\omega_{EV} = \frac{V_a}{\psi_m} \tag{2.10}$$

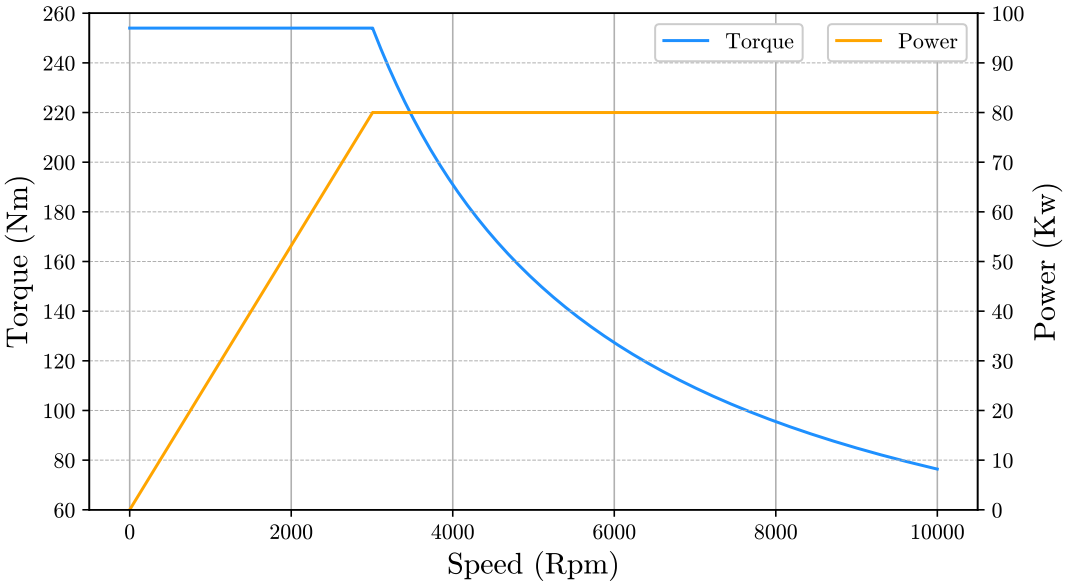


Figure 2.3: Synchronous Motor Characteristics.

2.3.2 Induction Motor

An induction motor works very similarly to a synchronous motor, except there is no current supplied to the rotor windings. The magnetic field produced by the stator interacts with the rotor windings and produces a different magnetic field in the rotor windings. Once again the interaction between the two magnetic fields causes the rotor to rotate [12]. Figure 2.4 shows the equivalent- and thevenin equivalent circuits of an induction motor.

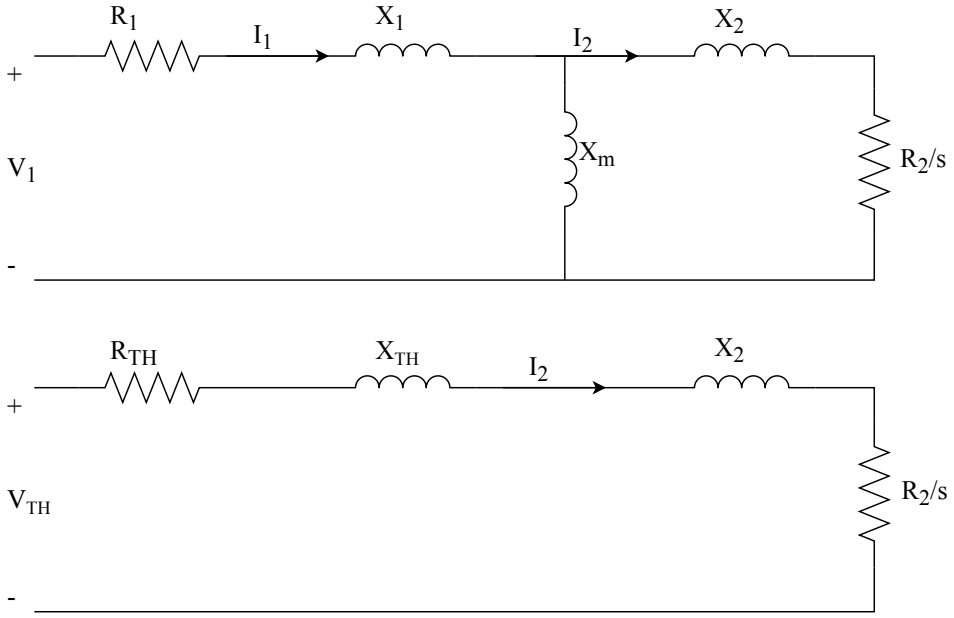


Figure 2.4: Induction Motor Equivalent Circuits. (Adapted from [14])

The torque developed by a induction motor can be determined by understanding the equivalent circuit and making use of Equation 2.11 through 2.13 [11]. Figure 2.5 shows the power and torque vs speed for a induction motor with parameters as seen in Table 2.2. An induction motor does not produce constant power nor constant torque.

$$T_{EV} = \frac{V_{TH}^2 \cdot R_2}{[(R_{TH} + R_2/s)^2 + (X_{TH} + X_2)^2] \cdot (s \cdot \omega_{sync})} \tag{2.11}$$

$$Z_{TH} = \frac{jX_m(R_1 + jX_1)}{R_1 + jX_1 + jX_m} \tag{2.12}$$

$$s = \frac{\omega_{sync} - \omega_{EV}}{\omega_{sync}} \tag{2.13}$$

- V_{TH} - Thevenin Equivalent Voltage (V).
- Z_{TH} - Thevenin Equivalent Impedance (Ω).
- R_{TH} - Thevenin Equivalent Resistance (Ω).
- X_{TH} - Thevenin Equivalent Reactance (Ω).
- R_1 - Stator Effective Resistance (Ω).
- R_2 - Referred Rotor Resistance (Ω).
- X_m - Magnetizing Reactance (Ω).
- s - Slip.

- ω_{sync} - Synchronous Speed $(m \cdot s^{-1})$.

Table 2.2: Induction Motor Specifications. [14]

Description	Value
R_1	0.461 Ω
R_2	0.258 Ω
X_1	0.510 Ω
X_2	0.756 Ω
X_m	30.74 Ω
ω_{sync}	1200 $m \cdot s^{-1}$

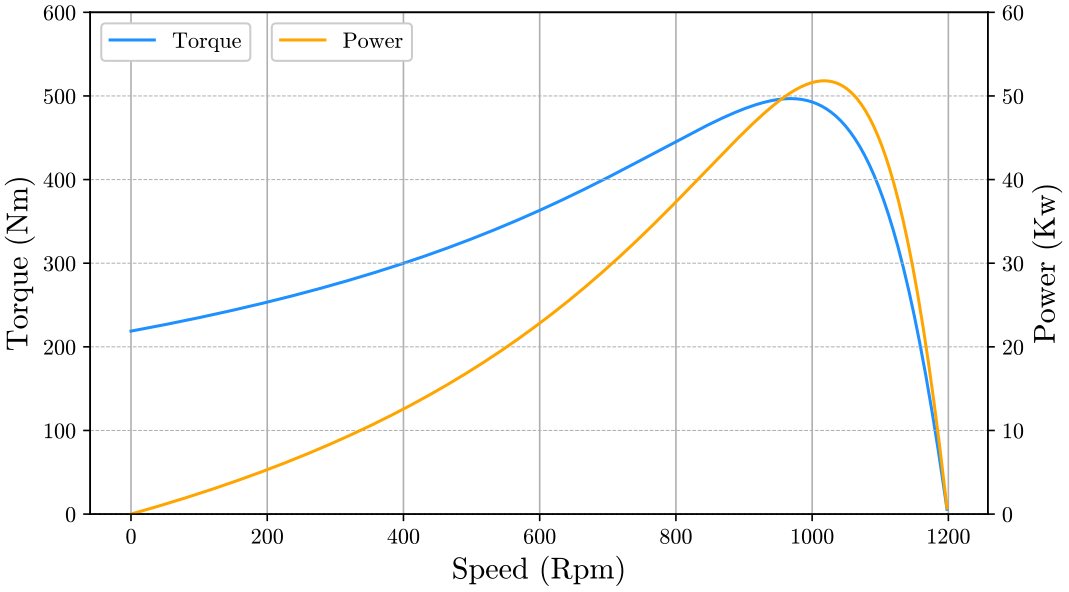


Figure 2.5: Induction Motor Characteristics.

2.4 Batteries

Batteries are arguably the most important part of an EV, as it provide the EV with energy. The battery of the EV is also one of the factors that will have the biggest effect on the range of the EV. But what is a battery? A battery consists of at least two cells that are joined together. The cells provide electrical energy by converting it from chemical energy. Cells consist of a cathode (positive electrode) and anode (negative electrode) which are joined together by an electrolyte. The chemical reaction between the electrolyte and the two electrodes generates the electrical energy [4]. The basic equivalent circuit of a

battery can be seen in Figure 2.6. The battery terminal voltage can be determined by Equation 2.14.

$$V_{bat} = E_{bat} - I_{bat} \cdot R_{int} \tag{2.14}$$

- V_{bat} - Battery Terminal Voltage (V).
- E_{bat} - Battery Internal Voltage (V).
- R_{int} - Battery Internal Resistance (Ω).
- I_{bat} - Battery Current (A).

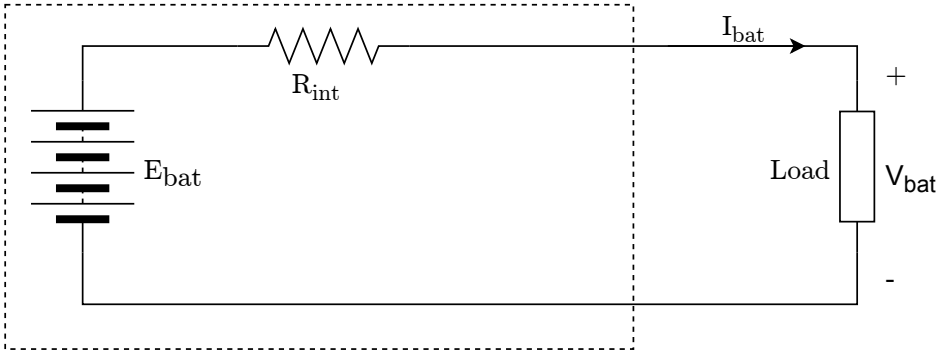


Figure 2.6: Battery Equivalent Circuit. (Adapted from [4])

To fully optimise the design of an EV in terms of battery choice, the battery with the highest specific energy will be superior. So in other words, the battery that has the highest Wh/kg will be the best.

2.4.1 Battery Topologies

There are several battery topologies currently in use, with Lead acid and Lithium-ion arguably being the most popular. A comparison between Lead acid and Lithium-ion batteries can be seen in Table 2.3. A short overview of these two battery types will follow.

Table 2.3: Battery Comparison [15]

	Flooded Lead Acid	**Sealed Lead Acid**	**Lithium-ion**
Specific Energy (Wh/kg)	30	40	150
Typical Cost (R/kWh)	1000	1800	9000
Life Cycles (% DoD^*)	1200 @ 50	1000 @ 50	1900 @ 80
Regular Maintenance	Yes	No	No
Maximum Operating Temperature ($°C$)	25	25	45

* Depth of Discharge - Battery discharge level compared to full capacity

2.4.1.1 Lead acid

Lead acid batteries are the oldest and most widely used battery topology and have been around for more than 100 years, invented in 1859 [16]. The electrodes of a fully charged Lead acid battery consists of one lead metal (Anode)- and one lead oxide (cathode) electrode. The electrolyte the electrodes are submerged in is sulphuric acid. When the the battery is discharged, both electrodes become lead sulphite and the electrolyte becomes water [16]. The overall reaction of a Lead acid battery is described by Equation 2.15 [4].

$$Pb + PbO_2 + 2H_2SO_4 \leftrightarrow 2PbSO_4 + 2H_2O \tag{2.15}$$

The advantages of Lead acid batteries are the low cost and reliability, where as the disadvantages thereof are the short life cycle and the low specific energy. Lead acid batteries can be divided into two subsections namely flooded and sealed [15]. There are three main differences between the two, and they are as follows:

- Flooded has to be in a upright position,
- Flooded has to be placed in a well ventilated room and
- Flooded needs routine electrolyte maintenance.

2.4.1.2 Lithium-ion

Lithium-ion batteries were first introduced to the public in the early 1990's. The cathode of a Lithium-ion battery consists of a lithiated metal oxide and the anode consists of a graphitic carbon. The electrolyte of Lithium-ion batteries are made of Lithium salts which are dissolved in organic carbonates. When a Lithium-ion battery is discharged, electrons flow from the anode to the cathode via the load, where Lithium-ions flows from the anode to the cathode via the electrolyte [16]. An example of the reaction at the cathode and anode is

described by Equations 2.16 and 2.17 respectively [17]. The advantages of Lithium-ion batteries are the high specific energy and long life cycle, where the main disadvantage is the high cost thereof.

$$LiCoO_2 \leftrightarrow 0.5Li^+ + 0.5e^- + Li_{0.5}CoO_2 \tag{2.16}$$

$$Li^+ + e^- + C_6 \leftrightarrow LiC_6 \tag{2.17}$$

2.4.2 Modelling

For the purposes of this project, the battery is simply modelled as en energy source with a certain capacity and this capacity is decreased as the EV uses energy along the route. The depth of discharge is assumed to be 100%, but can easily be changed by changing the available battery capacity within the software.

The coulombic efficiency of a battery refers to the amount of usable energy a battery has. The coulombic efficiency is a relationship between the charge- and discharge capacity of a battery. The coulombic efficiency can be calculated by Equation 2.18.[18] As well as for the depth of discharge, the coulombic efficiency is assumed to be 100% for this project. The coulombic efficiency can easily be changed as desired within the software, as discussed in Chapter 5.

$$CE = \frac{C_D}{C_C} \cdot 100 \tag{2.18}$$

- CE - Coulombic Efficiency (%).
- C_D - Discharge Capacity (kWh).
- C_D - Charge Capacity (kWh).

2.5 Sensitivity Analysis

There are various variables that will affect the range of the EV. Some of these variables can be controlled by the driver, where others are fixed and determined with the design of the EV. Environmental changes can also have an affect on the range of the EV. A few of the factors influencing the range of an EV will be investigated and discussed below. To investigate the different influences, simulations are done for a route that has a 0% incline and the EV is allowed to drive 80 km/h, with no power limitation implemented.

2.5.1 Tyre Pressure

Tyre pressure affects the range of an EV in the sense that tyre pressure will influence the coefficient of rolling resistance. A lower tyre pressure will increase the coefficient of friction where a higher tyre pressure will decrease the coefficient of friction. A study conducted by Suyabodha, A. [19] concluded that, compared to a base tyre pressure of 35 psi, a drop in 10 psi will cause the coefficient of friction to increase by 48.52%. An increase of 10 psi will cause the coefficient of friction to decrease by 13.46%. Table 2.4 shows the coefficient of friction for the different tyre pressures. In Figure 2.7 the influence of the tyre pressure on the range of the EV can be seen clearly. Flatter tyres can lead to the loss of about 40 km of range, and this will be even more at higher road inclines. It is thus important for the driver to closely monitor the tyre pressure, as it can have a significant effect on the range. For this project, it is assumed that the tyre pressure will stay constant.

Table 2.4: Coefficient of Friction [19]

Tyre pressure	**Coefficient of Friction**
25 *psi*	0.011139
35 *psi*	0.0075
45 *psi*	0.0064905

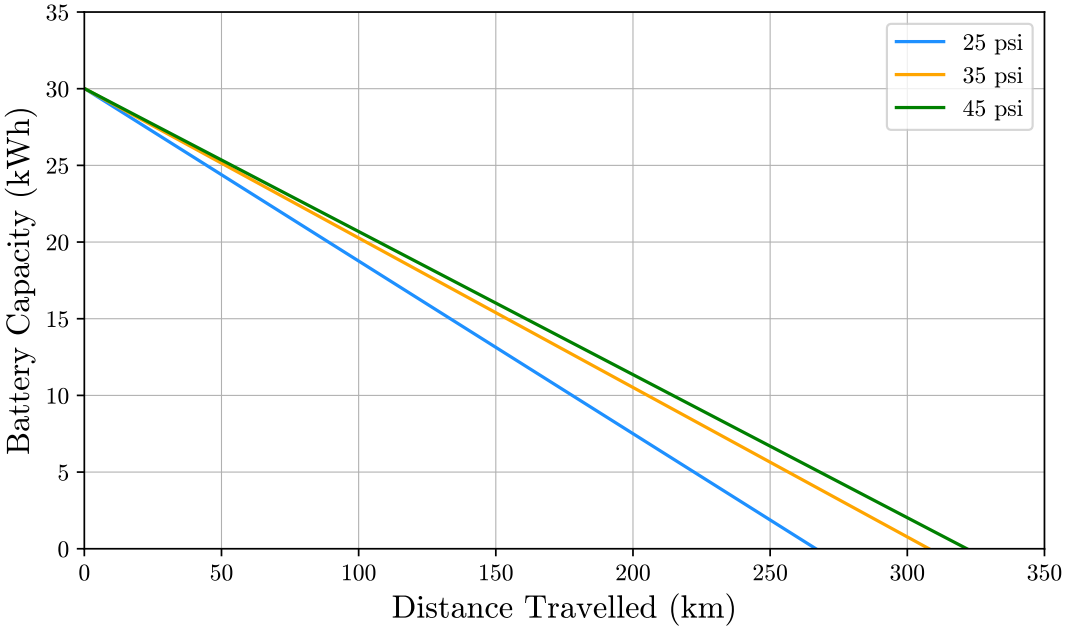


Figure 2.7: Tyre Pressure vs Range.

2.5.2 Ambient Temperature

Ambient temperature will have an effect on the range of an EV, as the ambient temperature will influence the air density [20]. Table 2.5 shows what the air density will be at different ambient temperatures. Figure 2.8 shows the effect that the ambient temperature will have on the range of the EV. The ambient temperature has a small but noticeable effect on the range, only about 10 km. For this project, the air density was kept constant, as it does not have a big influence on the range.

Table 2.5: Air Density [20]

Ambient Temperature	**Air Density**
$-10°C$	$1.342\ kg \cdot m^{-3}$
$15°C$	$1.225\ kg \cdot m^{-3}$
$45°C$	$1.172\ kg \cdot m^{-3}$

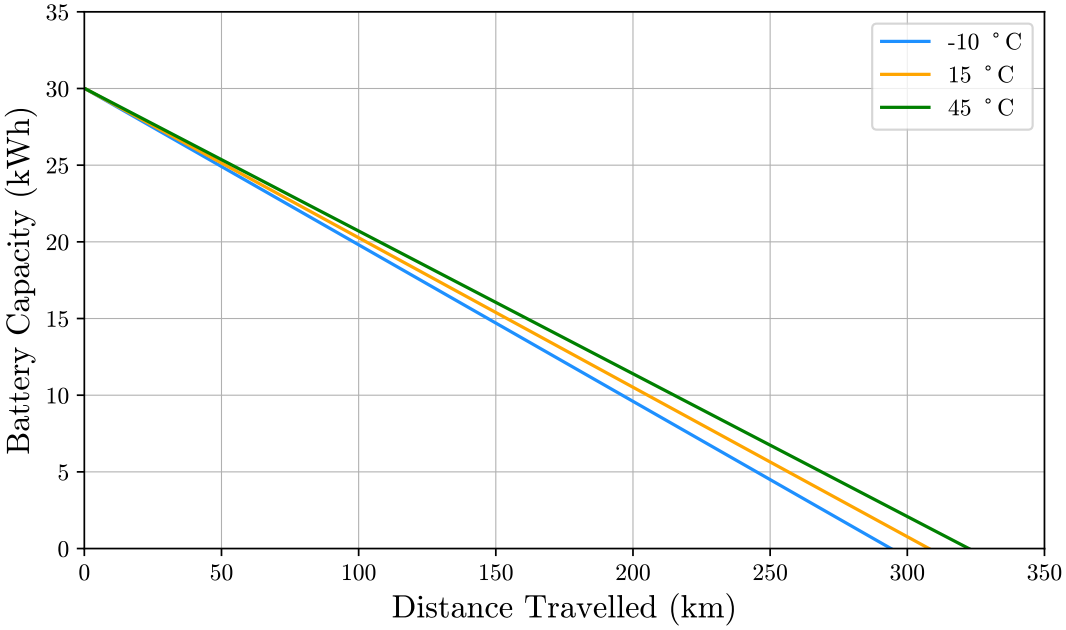


Figure 2.8: Ambient Temperature vs Range.

2.5.3 Aerodynamic Drag

Aerodynamic drag is one of the biggest forces an EV has to overcome in order to move forward. The drag coefficient will have a big influence on the aerodynamic drag force. The drag coefficient varies for each model of vehicle. To investigate the influence of the drag coefficient on the range of the EV, the drag coefficient of three different vehicles will be used in simulation, as seen in Table 2.6. Figure 2.9 shows the impact of the drag coefficient on the range. The drag coefficient has a big effect on the range of the EV, averaging at more than 100 km. For this simulation, it was assumed that it is the exact same vehicle, but with different drag coefficients. The driver of the EV has no control over the drag coefficient, this can only be altered in the design process of the EV.

Table 2.6: Drag Coefficient [21], [22]

Vehicle	Drag Coefficient
Nissan Leaf	0.28
Fiat Turbina	0.14
Hummer H2	0.53

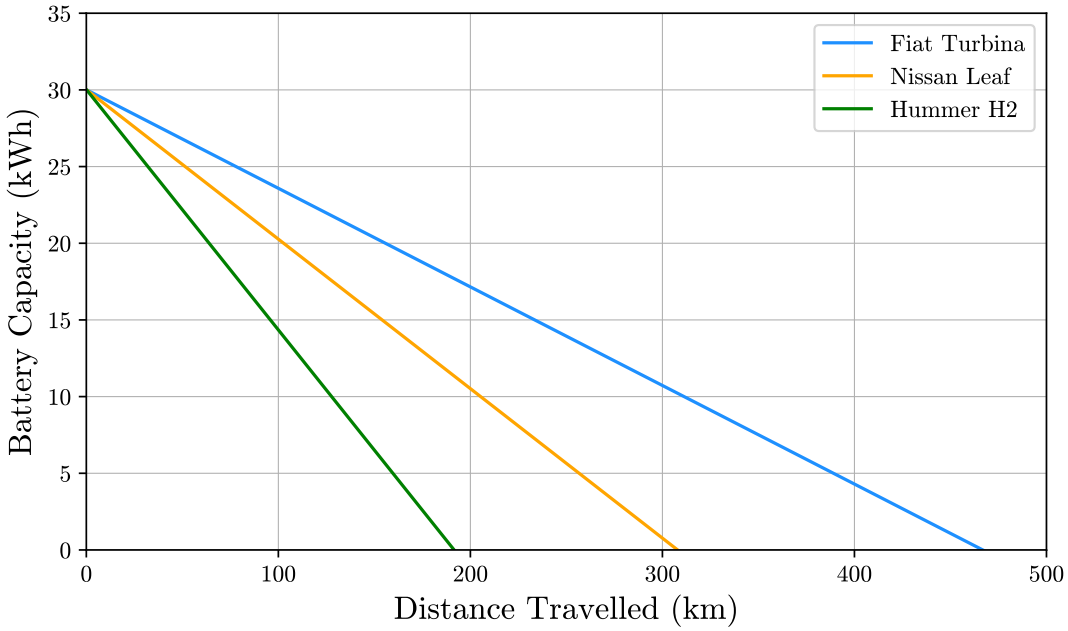


Figure 2.9: Drag Coefficient vs Range.

2.5.4 Road Angle

The angle of the road will affect the hill climbing force needed to move the EV. A steeper road will require more power from the EV to move, compared to a less steep road. Figure 2.10 shows how the incline of the road will affect the range of the EV. A road with an incline of 10% states that over the coarse of 100m, the road elevates 10m. The road incline has a very big effect on the range of the EV, and is therefore important for the driver to do careful route planning to avoid mountainous roads such as passes, as far as possible.

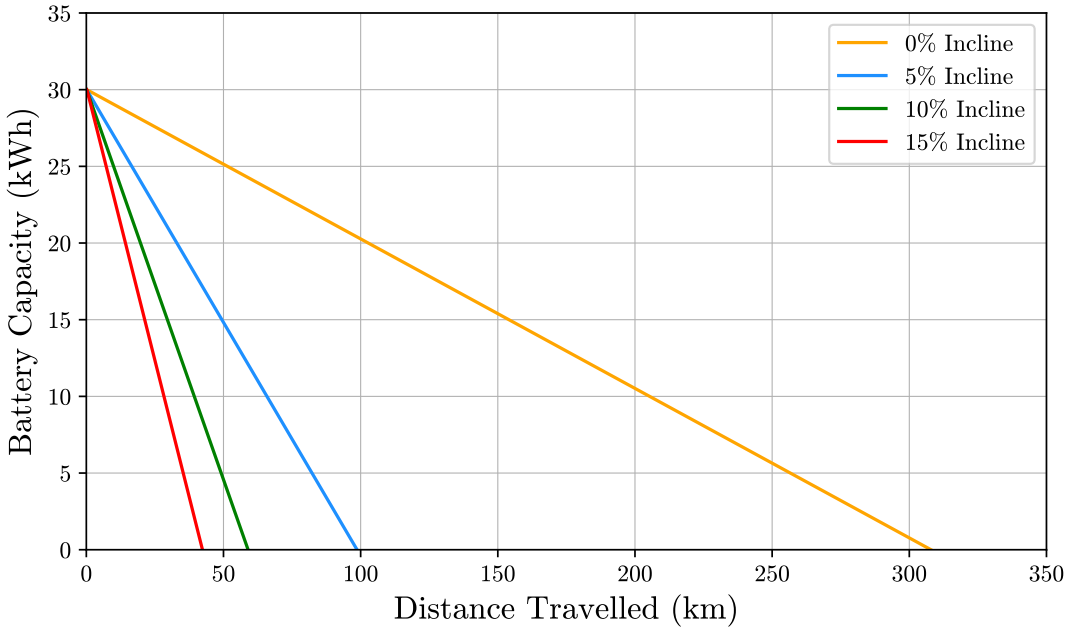


Figure 2.10: Road Incline vs Range.

2.5.5 Battery Technology Development

As battery technology develops, batteries continue to get a higher specific energy. This means that for the same battery size, the weight will be less. Or if you prefer to keep the weight as is, a battery with a much higher capacity can be used. The 2017 Tesla Model S has a 100 kWh battery that weighs 453 kg. This equates to a specific energy of 220 Wh/kg, compared to the 102 Wh/kg battery that is in the 2016 Nissan Leaf [23]. Table 2.7 shows the battery size vs battery weight for 3 different cases used for comparison. From Figure 2.11 it is clear to see that double the battery capacity is much more advantageous compared to half the weight of the original battery capacity.

Table 2.7: Battery Weight

Battery Size	**Battery Weight**
30 kWh	294 kg
30 kWh	136 kg
65 kWh	294 kg

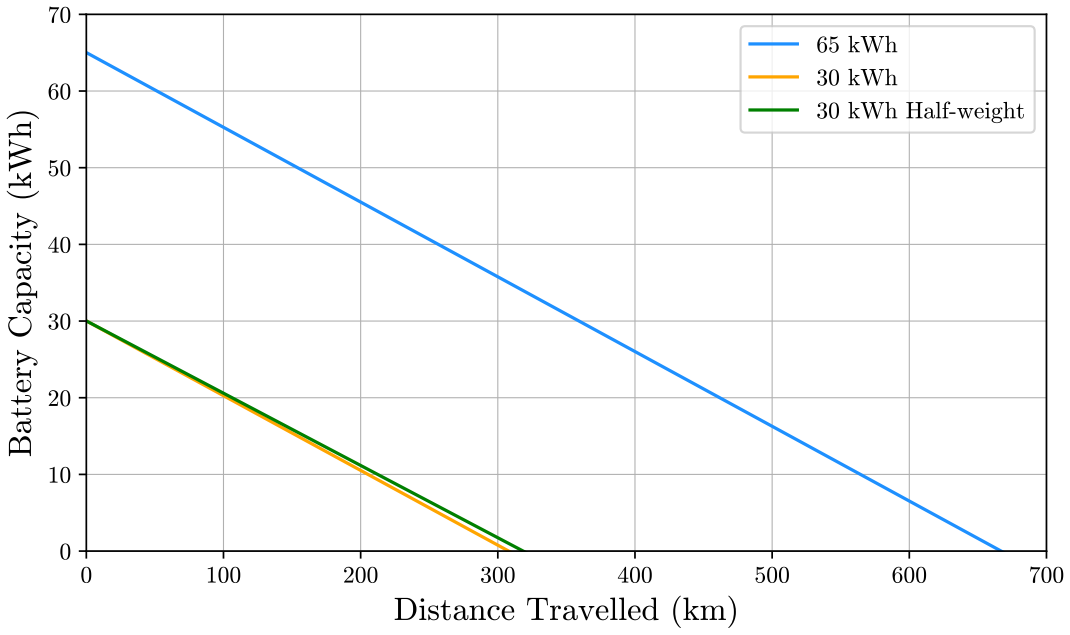


Figure 2.11: Battery Development vs Range.

Chapter 3

Solar Modelling

3.1 Introduction

The sun is the resource that ensures that the earth maintains the correct temperature, makes it possible for plants to photosynthesise, controls our weather and provides the energy to make photovoltaic (PV) systems possible. According to Einstein's theory of special relativity, as seen in Equation 3.1, approximately 4 million tons of mass is converted into energy each second [24]. To make this light energy useful, it has to be converted into electrical energy and this is where solar panels come in.

$$E = mC^2 \tag{3.1}$$

- E - Energy (J).
- m - Mass (Pascal).
- C - Speed of Light (m/s).

3.2 Irradiance

The energy that the sun emits, is called solar radiation. The difference between irradiation/solar radiation and irradiance, is that irradiation refers to the energy received per area, where irradiance refers to power received per area. Say for instance there are only clear-sky days in the year, the amount of irradiance that a fixed panel will receive, will not be the same year round. This is due to the position of the earth relative to the sun, which is constantly changing. The earth orbits the sun in a spherical path, which means that the distance from the sun to the earth will not be constant during a year [25].

The rotational axis of the earth is tilted at 23.45°. This is responsible for the different seasons we experience and will effect the incidence angle of the suns'

rays that hits the earth. This will in turn have a big influence on the amount of irradiance the panel will receive [24]. Irradiance can be broken down into 3 main categories, namely: Global Horizontal Irradiance (GHI), Direct Normal Irradiance (DNI) and Diffuse Horizontal Irradiance (DHI). The intensity of each of these irradiances differ, and an example of this for a clear-sky day can be seen in Figure 3.1. These irradiances are interdependent and will be discussed below.

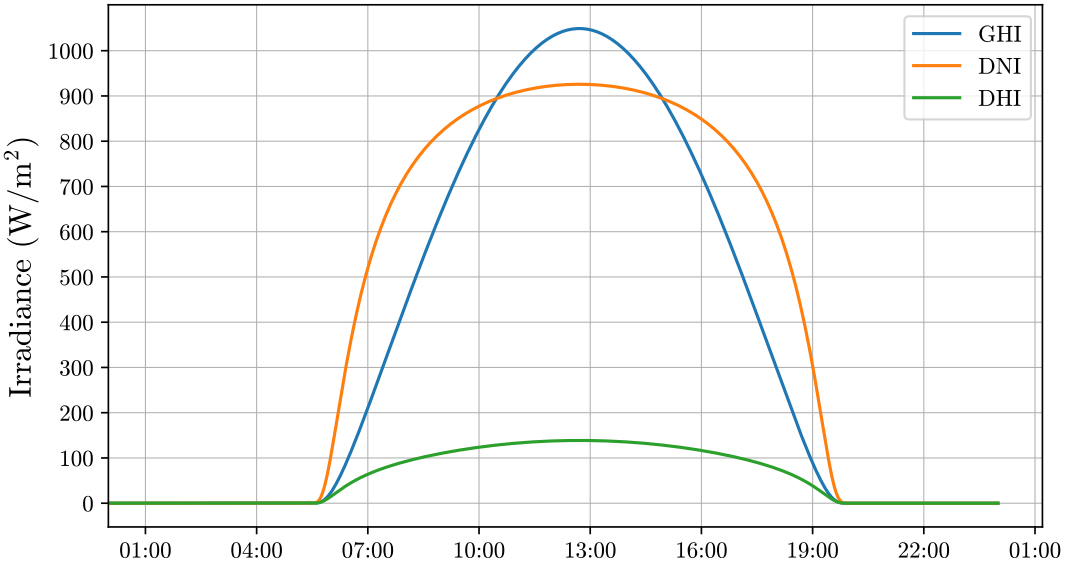


Figure 3.1: Clearsky GHI vs DNI vs DHI.

3.2.1 Global Horizontal Irradiance

The global horizontal irradiance is defined as the total amount of irradiance that is received on a surface that is held completely horizontal. GHI consists of DNI as well as DHI [26]. As Chapter 4 will cover GHI prediction, the GHI will be known, where as DNI and DHI will have to be derived from the GHI.

3.2.2 Direct Normal Irradiance

The direct normal irradiance is defined as the total amount of irradiance that is received on a surface that is always held perpendicular to the incoming rays of the sun [27]. The DNI can be determined by making use of the pvlib library from Python. Pvlib determines the DNI by implementing Equations 3.2 through 3.6, and is discussed below [28] [29].

$$DNI = ER \cdot X(Kt^1, \theta_z, W, \Delta Kt^1) \tag{3.2}$$

$$Kt^1 = \frac{Kt}{(1.031 \cdot \exp(-1.4/(0.9 + 9.4/m_a)) + 0.1)} \tag{3.3}$$

$$\Delta Kt^1 = \frac{1}{2} \cdot (|Kt_i^1 - Kt_{i+1}^1| + |Kt_i^1 - Kt_{i-1}^1|) \tag{3.4}$$

$$W = \exp(0.07 \cdot T_d - 0.075) \tag{3.5}$$

$$Kt = \frac{GHI}{ER \cdot \cos\theta_z} \tag{3.6}$$

- ER - Extraterrestrial Radiation (W/m^2).
- $X(Kt^1, \theta_z, W, \Delta Kt^1)$ - Coefficient Function.
- Kt - Global Horizontal Transmittance.
- Kt^1 - Clearness Index.
- $\Delta Kt^{'}$ - Stability Index.
- θ_z - Solar Zenith Angle ($^\circ$).
- W - Atmospheric Precipitable Water (cm).
- m_a - Air Mass.
- $Kt_i^1, Kt_{i\pm1}^1$ - Current Kt^1 and Kt^1 at Next/Previous Hour.
- T_d - Dew-Point Temperature ($^\circ C$).

The above mentioned coefficient function consists of 4 parameters, as can be seen in Equation 3.2. The coefficient is obtained from 2 lookup tables. The four parameters within the coefficient function are divided into so called bins. To divide the parameters into these bins, Table 3.1 is seen as the first lookup table [29]. This second lookup table is four-dimensional and consists of a 6 $\times$ 6 $\times$ 5 $\times$ 7 matrix as discussed in [28] and [29] . Once the parameters are divided into bins, they will have a corresponding value within the second lookup table. For this application, ER is seen as the product of the solar constant and the distance from the earth to the sun on the particular day [30].

Table 3.1: Bin Classification Lookup Table

Bin	$\boldsymbol{Kt^{'}}$	$\boldsymbol{\theta_z}$	**W**	$\boldsymbol{\Delta Kt^{'}}$
1	0 - 0.24	0 - 25	0 - 1	0 - 0.015
2	0.24 - 0.4	25 - 40	1 - 2	0.015 - 0.035
3	0.4 - 0.56	40 - 55	2 - 3	0.035 - 0.07
4	0.56 - 0.7	55 - 70	3 - ∞	0.07 - 0.15
5	0.7 - 0.8	70 - 80	0 - ∞ *	0.15 - 0.3
6	0.8 - 1	80 - 90		0.3 - 1
7				0 - 1 *

* Used respectively when W and $\Delta Kt^{'}$ are not available

3.2.3 Diffuse Horizontal Irradiance

Diffuse horizontal irradiance is defined as irradiance that does not come directly from the sun, but is scattered from aerosols, dusts and particles in the atmosphere. DHI comes equally from all directions. So for a clear day, the GHI will consist mostly out of DNI, where as on a cloudy day the GHI will consist mostly out of DHI [31]. As the GHI and DNI is known, the DHI can be calculated by the relationship between the three as described by Equation 3.7 [32].

$$DHI = GHI - DNI\cos(\theta_z) \tag{3.7}$$

3.2.4 Total In-plane Irradiance

The total in-plane irradiance is seen the total amount of irradiance that is received on a tilted surface. This includes direct irradiance, diffuse irradiance as well as reflected irradiance [33]. For this thesis, it is important to be able to determine the in-plane irradiance, as the tilt of the solar panel will constantly change as the car drives along the route. The in-plane irradiance is needed to accurately predict the amount of power the panel will provide to the vehicle. The total in-plane irradiance can be determined by making use of pvlib. Pvlib implements Equations 3.8 through 3.11, and is discussed below [34]. The code for these calculations can be seen in Appendix 7.6

$$POA = POA_{direct} + POA_{diffuse} \tag{3.8}$$

- POA - Total In-plane Irradiance (W/m^2).
- POA_{direct} - Total In-plane Direct Component (W/m^2).
- $POA_{diffuse}$ - Total In-plane Diffuse Component (W/m^2).

The direct component can be determined by Equation 3.9 and the diffuse component can be determined by Equation 3.10 [34].

$$POA_{direct} = DNI \cdot \cos(\theta_{AOI}) \tag{3.9}$$

$$POA_{diffuse} = DHI \cdot \left(\frac{1+\cos\beta_t}{2}\right) + GHI \cdot \left(\frac{1-\cos\beta_t}{2}\right)\rho \tag{3.10}$$

- ρ - Surface Albedo.
- β_t - Solar Panel Tilt Angle (°).

The angle of incidence is defined as the angle between a line perpendicular to the panel and a line pointing directly at the sun, as seen in Figure 3.2 [24]. So

for a panel that is mounted flat on the ground, the angle of incidence will be equal to the solar Zenith angle. The angle of incidence can be determined by Equation 3.11 [35].

$$\cos(\theta_{AOI}) = \cos^{-1}\left[\cos\beta_t \cos\theta_z + \sin\beta_t \sin\theta_z \cos(\psi - \psi_s)\right] \tag{3.11}$$

- θ_{AOI} - Angle of Incidence (°).
- ψ - Solar Azimuth Angle (°).
- ψ_s - Surface Azimuth Angle (°).

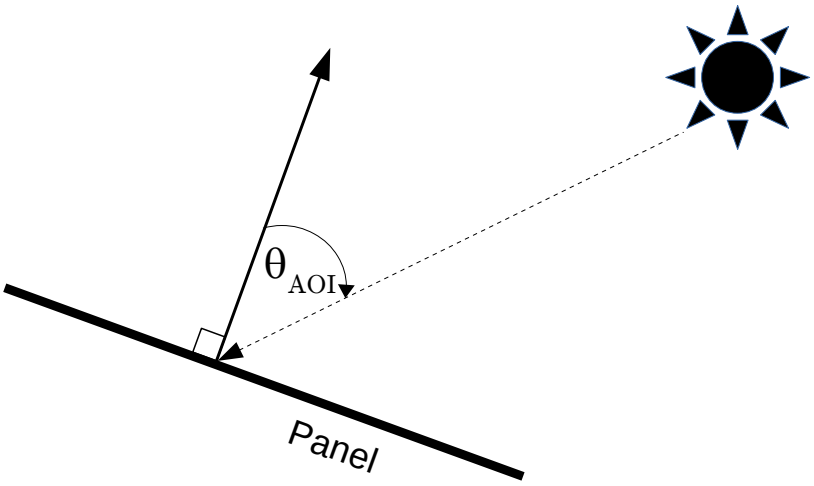


Figure 3.2: Angle of Incidence. (Adapted from [36])

3.3 Solar Angles

The sun follows a path from East to West. The path of the sun is not the same year round, it is dependant on the time of year (the season). The path can however be defined by two solar angles, namely the solar Zenith and -Azimuth. So in other words, with the solar Zenith and -Azimuth known, the exact position of the sun in the sky can be determined [37]. This is necessary in order to compute the total in plane irradiance that the solar panel will receive at any given point in time, as discussed in Chapter 3.2. All of the equations discussed in this section, are applicable to the Southern hemisphere. There will be minor differences if the calculations are done for the Northern hemisphere.

The solar Zenith angle describes the vertical position of the sun in the sky. It is measured from a line perpendicular towards your position on earth, to the sun (in degrees). The solar Zenith angle (θ_z) can be better understood by the aid

of Figure 3.3. The solar Azimuth angle (ψ) indicates the horizontal position of the sun in the sky. It is measured in degrees with respect to true north, and can be seen in Figure 3.3 [25]. The Zenith and Azimuth are determined by making use of pvlib. The equations used by pvlib are discussed below.

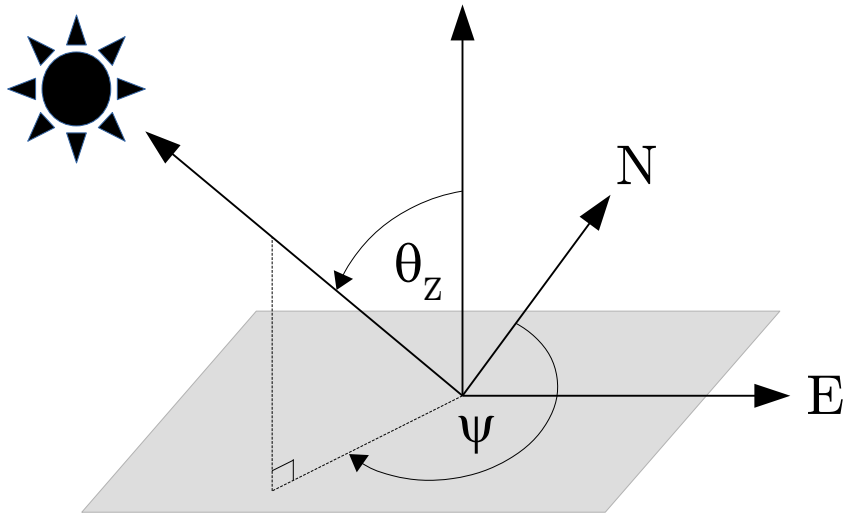


Figure 3.3: Zenith vs Azimuth. (Adapted from [38])

To determine the exact solar Zenith and -Azimuth, there are a few variables that have to be known: the latitude, the number of the day and the current time. Latitude is described as your angular distance from the Equator. Values that are to the North of the equator are seen as positive. Longitude is described as your angular distance from the Greenwich Meridian and is considered as positive to the East thereof [25]. Figure 3.4 shows a visual representation of the difference between latitude and longitude.

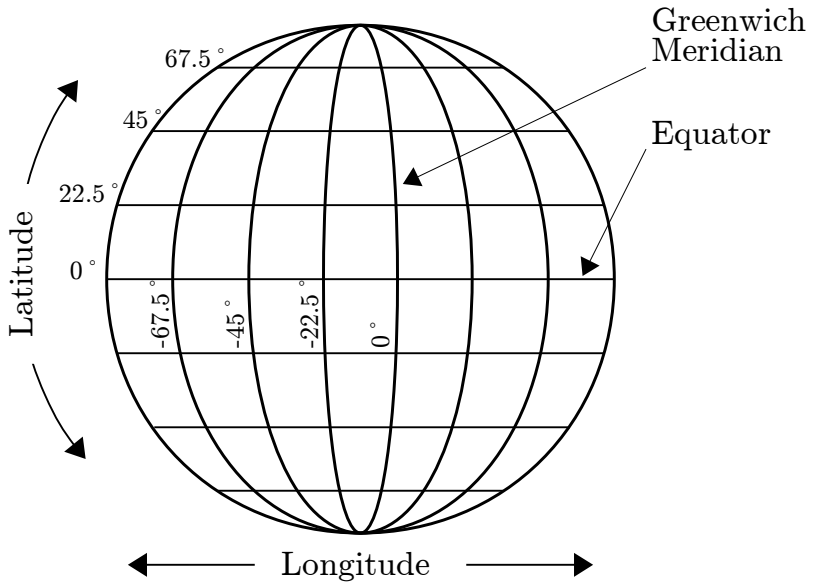


Figure 3.4: Latitude vs Longitude. (Adapted from [39])

January the first is seen as day number 1 for all solar calculations. The current time is needed to determine the so called hour angle. The hour angle shows how many degrees the earth has to rotate in order to reach solar noon (12:00 AM solar time). As the earth rotates 360 degrees in 24 hours, 1 hour is equal to 15 degrees of rotation [24]. The hour angle is seen as negative in the morning, and is determined by Equation 3.12.

$$H = (LST - 12) \cdot 15^{\circ} \tag{3.12}$$

To determine the hour angle, local time has to be converted to Local Solar Time (LST). The first step in doing this, is to determine the Local Standard Time Meridian (LSMT). The LSMT can be determined by Equation 3.13, where ΔT_{UTC} is the difference between local time and Universal Coordinate Time (UTC) [24].

$$LSMT = 15^{\circ} \cdot \Delta T_{UTC} \tag{3.13}$$

The next step is to determine the Equation of Time (ET). The equation of time accounts for the difference between a day and a solar day, as the length of a solar day varies through the year. A solar day is defined as solar noon to solar noon. The equation of time can be determined by Equation 3.14, where n is the number of the current day.

$$ET = 9.87\cdot\sin\left[\frac{2\cdot 360}{365}(n-81)\right] - 7.53\cdot\cos\left[\frac{360}{365}(n-81)\right] - 1.5\cdot\sin\left[\frac{360}{365}(n-81)\right] \tag{3.14}$$

The LST is defined by Equation 3.15, where TC is the Time Correction Factor (TC). The TC accounts for the difference between the local time zone and LST, due to the longitudinal position. The TC can be determined by Equation 3.16 [40].

$$LST = LocalTime + \frac{TC}{60} \tag{3.15}$$

$$TC = 4\cdot(Longitude - LSMT) + ET \tag{3.16}$$

The solar Zenith and solar Azimuth angles can now be determined with Equation 3.17 and Equation 3.18 [25]. The declination angle can be expressed by Equation 3.19.

$$\cos\theta_z = \sin L\sin\delta + \cos L\cos\delta\cos H \tag{3.17}$$

$$\sin\psi = \frac{\cos\delta\sin H}{\cos(90-\theta_z)} \tag{3.18}$$

- L - Latitude ($^\circ N$).
- δ - Declination Angle ($^\circ$).
- H - Hour Angle ($^\circ$).

$$\delta = 23.45^\circ\sin\left[\frac{360}{365}(n+284)\right] \tag{3.19}$$

There is one more angle that is needed to calculate the total in plane irradiance, and that is the surface Azimuth angle. The surface Azimuth is defined as the angle between a line perpendicular to the surface of the panel, and North. So if a solar panel is pointed directly East, the surface Azimuth will be 90° [25]. To clarify the difference between solar- and surface Azimuth: For a solar panel that is fixed, the surface Azimuth will stay constant, where as the solar Azimuth will be different for every hour of every day. The surface Azimuth is included in the road profile data, which is the input of the system. The calculation of this angle will be discussed in Section 5.

3.4 Single Diode Model

Solar cells are made of semiconductor materials, of which crystalline silicon is the most commonly used. The crystalline silicon is doped with so called impurities, in order to create a p-n junction. This p-n junction has a N-type semiconductor on the one side of the cell and a P-type semiconductor on the other side of the cell. The N-type semiconductor has free electrons with negative charge, where the P-type semiconductor has free openings with a positive charge. When the solar cell absorbs photons from the sun, so called hole-electron pairs are created. The n-side of the cell builds up electrons and the p-side of the cell builds up holes. Thus, if a conducting wire connects the n-side and the p-side of the cell, electrons will flow from the n-side to the p-side, as seen in Figure 3.5 [41]. Each cell provides about 0.5 V when it is submitted to direct sunlight.

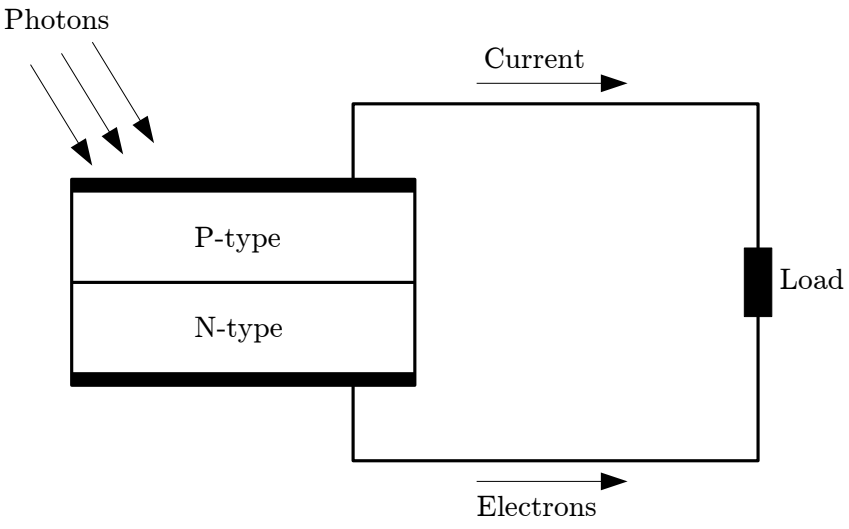


Figure 3.5: Basic Solar Cell. (Adapted from [24])

Now that the basic working of a solar cell is known, it is important to distinguish between a PV cell, PV module, and a PV array. A PV module consists of several PV cells that are connected in series, parallel or series-parallel as seen in Figure 3.6. These connected cells are contained within a frame, with a transparent cover which is usually made of glass. This protects the cell from damaging environmental elements. The amount of cells within a PV panel will determine the maximum DC output of the panel [42]. A PV panel is also fitted with bypass diodes, as shaded cells cannot produce any current. The bypass diode is a precautionary method for preventing hotspots on the panel,

which can lead to permanent damage, as well as power losses when the panel is subject to partial shading [43]. The bypass diode solves this problem by simply diverting the current through the diode instead of the shaded cells. A PV array is simply several PV panels that are connected in series and parallel to provide the desired DC output power.

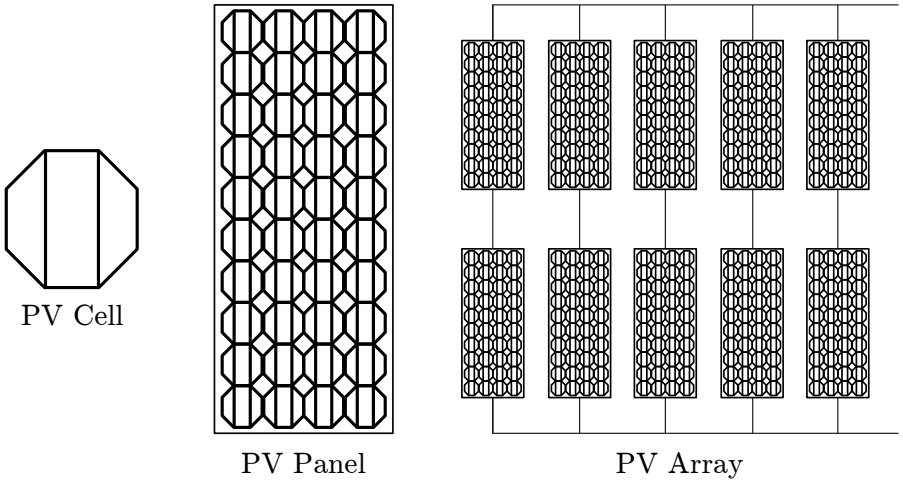


Figure 3.6: PV Cell, -Module and -Array. (Adapted from [42])

Now that the basic workings of solar power are known, the next step is to model a solar panel. There are currently 2 main circuit representations to model a solar cell, namely the single- and double diode models. For the purpose of this project, it is decided to implement the single diode model. The most simplistic way to model an ideal solar cell, is by putting a diode in parallel with a current source, as seen in Figure 3.7. The problem with this ideal model, is that it becomes very inaccurate when it is subjected to environmental variations [44]. To improve these inaccuracies, two resistors are added. One is added in series and the other one in parallel. The series resistor, Rs, is put into the circuit to take voltage drops and internal losses into account [45]. The parallel resistor, Rp, is put into the circuit to take into account what happens when the diode is reverse biased, i.e. the leakage current flows into ground [45].

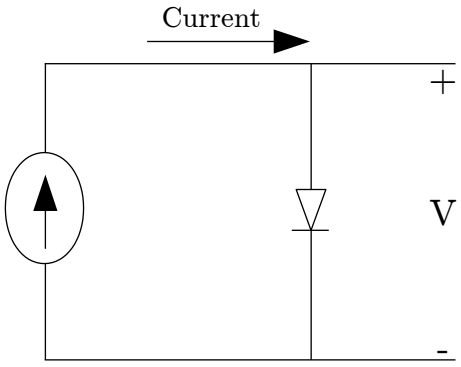


Figure 3.7: Ideal Solar Cell. (Adapted from [44])

3.4.1 Design Theory

Figure 3.8 shows the practical model used for the modelling of a solar cell. The model can also be referenced as the five parameter model. It is called the five parameter model as there are five key unknowns that have to be solved in order to successfully model the cell, namely [45]:

- R_s - Series Resistance (Ω),
- R_p - Parallel- or Shunt Resistance (Ω),
- I_p - Photocurrent (A),
- I_o - Diode Saturation Current (A) and
- α - Diode Ideality Factor.

The output current of the circuit in Figure 3.8, can be described by Equation 3.20 [46]. V_t refers to thermal voltage equivalent, and can be determined by using Equation 3.21 [46].

$$I = I_p - I_o \cdot \left[exp\left(\frac{V + IR_s}{\alpha V_t} \right) - 1 \right] - \left[\frac{V + IR_s}{R_p} \right] \tag{3.20}$$

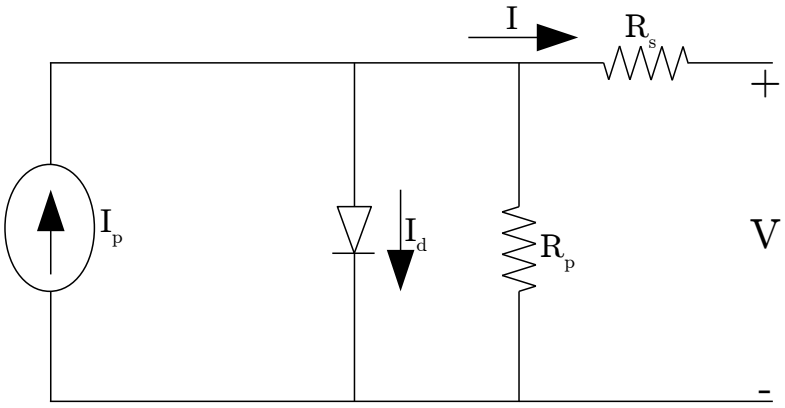


Figure 3.8: Single Diode Model. (Adapted from [44])

$$V_t = \frac{N_s k T_{cell}}{q} \tag{3.21}$$

- N_s - Number of cells connected in series.
- k - Boltzmann Constant ($J \cdot K^{-1}$).
- T_{cell} - Operating temperature of the solar cell ($°C$).
- q - Electron Charge (C).

The photocurrent needed to complete Equation 3.20, can be determined by Equation 3.22 [45]. As previously mentioned, the photocurrent is the current that is generated from the solar cell. It is dependant on the ambient temperature as well as the amount of irradiance it receives. Both the temperature and irradiance are taken into account when using Equation 3.22.

$$I_p = \left[I_{pn} + K_i \cdot \left(T_{cell} - T_n \right) \right] \cdot \frac{G}{G_n} \tag{3.22}$$

- I_{pn} - Photocurrent at Standard Test Conditions (STC) (A).
- K_i - Short circuit temperature coefficient.
- T_n - Temperature at STC ($25°C$).
- G - Irradiance (W/m^2).
- G_n - Irradiance at STC (1000 W/m^2).

The operating temperature of a solar panel is dependant on various factors, with the main factors being the ambient temperature and the windspeed. It is important to take the windspeed into consideration as it will lower the

operating temperature of the panel. A lower operating temperature will lead to an increase in efficiency. As the solar panel will be mounted on the roof of the vehicle, it will be subjected to moderate to high windspeed. One first have to determine the operating temperature of the panel itself, before determining the operating temperature of the cell. The operating temperature of the module can be determined by Equation 3.23 [47].

$$T_{mod} = POA \cdot (exp^{a+b \cdot WS}) + T_{amb} \tag{3.23}$$

- T_{mod} - Module Operating Temperature ($^{\circ}C$).
- a - Empirically Calculated Coefficient.
- b - Empirically Calculated Coefficient.
- WS - Windspeed (m/s).
- T_{amb} - Ambient Air Temperature ($^{\circ}C$).

For Equation 3.23, a and b were empirically calculated by [47]. Coefficient a refers to the upper limit of the module temperature, and is equal to -2.98. Coefficient b refers to the rate at which the windspeed causes a drop in the module temperature, and is equal to -0.0471. The cell temperature can now be determined by Equation 3.24, where ΔT_{emp} is a empirically calculated coefficient. ΔT_{emp} refers to the temperature difference between the module back plate and the cell at G_n and is equal to 1 $^{\circ}C$ for this application [47].

$$T_{cell} = T_{mod} + \frac{POA}{G_n} \cdot \Delta T_{emp} \tag{3.24}$$

The diode saturation current can be expressed by Equation 3.25, where E_g is the band gap energy of the semiconductor and I_{on} is the diode saturation current at STC [44].

$$I_o = I_{on}\left[\frac{T_n}{T}\right]^3 exp\left[\frac{qE_g}{\alpha k} \cdot \left(\frac{1}{T_n} - \frac{1}{T}\right)\right] \tag{3.25}$$

The last unknown to solve Equation 3.20, is the diode saturation current at STC (I_{on}). I_{on} can be solved by Equation 3.26 [45].

$$I_{on} = \frac{I_{scn}}{\left[exp\left(\frac{V_{ocn}}{\alpha V_{Tn}}\right) - 1\right]} \tag{3.26}$$

- I_{scn} - Short Circuit Current at STC (A).
- V_{ocn} - Open Circuit Voltage at STC (V).
- V_{Tn} - Thermal Voltage Equivalent at STC (V).

3.4.2 IV Curves

To complete the modelling of the solar panel, all of the five parameters have to be known in order to solve Equation 3.20. It is possible to calculate these parameters manually, however it is a substantial amount of equations to work through. For the purpose of this project, there is a much simpler solution, namely pvlib. The algorithms to develop the pvlib database, is derived from models from several peer-reviewed publications, as discussed above. Pvlib allows you to retrieve the model parameters of your chosen solar panel. This information is retrieved from NRELs (National Renewable Energy Laboratory) SAM (System Advisor Model) website.

The Canadian Solar CS6X-350-FG solar panel was chosen for the modelling of this thesis, and the specifications thereof can be seen in the datasheet [48]. This was chosen as it is a high efficiency poly-crystalline panel. It contains 72 individual cells, with a rated maximum power output of 350 W. The parameters of the chosen panel can be seen in Table 3.2. The input parameters for the pvlib algorithm are as follows: surface tilt, surface azimuth, irradiance and the cell temperature.

Table 3.2: Solar Panel Parameters.

Parameter	Datasheet Value	Modelled Value
Nominal Power	350 W	350 W
Voltage at Maximum Power	38.1 V	38.3 V
Current at Maximum Power	9.21 A	9.13 A
Open Circuit Voltage	46.2 V	46.6 V
Short Circuit Current	9.79 A	9.67 A
Cells	72(6 by 12)	N/A
Panel Weight	27.5 kg	N/A

Figure 3.9 shows the simulated I-V curves for 5 different irradiance levels, at a constant cell temperature of 25°C. To review the accuracy of the model that is used, the parameters from the data sheet are compared to the modelled values and can be seen in Table 3.2. It is concluded that these values match the values from the datasheet very closely, and the model is deemed accurate enough for the purposes of this thesis. The five parameters of the so called five parameter model can be seen in Table 3.3. When implementing this solar panel into the vehicle subsystem, it is important to take the weight of the panel into consideration in order to accurately review the contribution of the panel to the range of the vehicle.

Table 3.3: Module Parameters.

Parameter	Value
R_s	0.292 Ω
R_p	752.63 Ω
I_p	9.674 A
I_o	162 pA
α	1.88

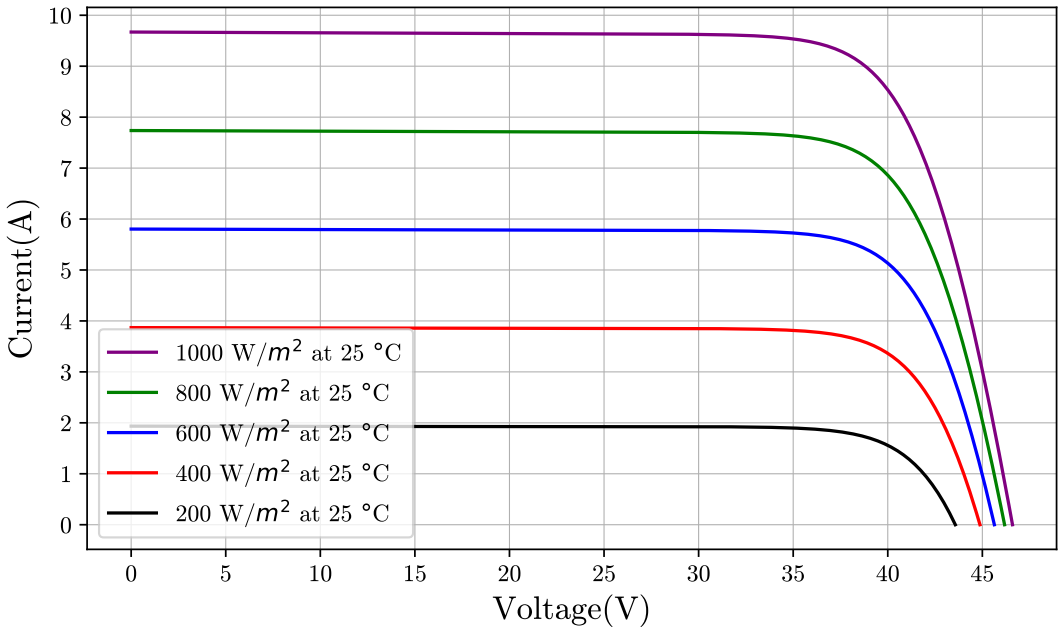


Figure 3.9: I-V curves at various irradiance levels.

3.5 PV Losses

For small applications, PV panels have a relative low yield and it is therefore important to minimize the losses where possible [49]. As for the case in this project, where only 2 PV panels are used, the losses have to be taken into consideration where necessary to calculate accurate power yield. The modelling of the system is done in such a way that the losses can easily be taken in- or not be taken into consideration, within the developed software discussed in Chapter 5. There are various losses with regards to PV yield, and will be discussed in short below.

3.5.1 Shading Losses

Shading losses are one of the most important factors to take into account when calculating power yield. For a PV panel that consists of 4 parallel strings, as seen in Figure 3.6, if only the the top corner of the panel is shaded, the power output of the panel will be reduced by 25% [50]. This is due to the bypass diode diverting the current away from the shaded parallel string. For this project, shading losses will not be taken into consideration, as it will not be possible to model the shading while the EV drives. Highways are regularly maintained and it is seldom for trees to overhang on the road, and thus it is justified to ignore the shading losses as the EV drives. For the case where the EV is parked, the judgement of the driver on where to park the EV, will lead to the panels not to be shaded where the EV is parked, hence no shading losses will occur.

3.5.2 Dust Losses

Dust losses, is as the name states, losses in PV yield to the PV panels being covered by dust. In extreme cases, power losses due to dust can be as high as 15%.[51] To mitigate power losses due to dust, it is important that the EV is cleaned regularly, or at least the PV panels. Due to the driver being instructed on regular cleaning of the PV panels, dust losses will not be taken into consideration.

3.5.3 DC Cable Losses

Another loss to take into consideration is the losses in the cables that connects the PV panels to the inverter. These losses are due to the resistance of the cabling and can be determined by Equation 3.27 [52]. The resistance of a cable is determined by Equation 3.28 [53]. The cable connected to the PV panels has a cross-sectional area of 4 mm^2, as stated in the datasheet. Copper conductors generally have a resistivity of $1.77 \times 10^{-8} \Omega m$ [54].

$$P_{DC} = 2 \cdot I^2 R_{DC} \tag{3.27}$$

$$R_{DC} = \frac{\rho_{cable} \cdot L_{cable}}{A_{cable}} \tag{3.28}$$

- P_{DC} - DC Cable Loss (W).
- R_{DC} - Cable Resistance (Ω).
- ρ_{cable} - Conductor Resistivity (Ωm).
- L_{cable} - Cable Length (m).
- A_{cable} - Cable Cross-sectional Area (m^2).

For the application in this project, the cabling will not be longer than 4 m for both PV panels. Keeping in mind that the current at the maximum power point of the Chosen PV panels is 9.21 A, the DC cable losses can be determined. Using Equation 3.27 and 3.28, the losses are calculated as 3 W, which equates to a 0.85% loss in power. This is so small, that the DC cable losses are deemed negligible for this project.

3.5.4 Inverter Losses

Both the PV panels and the battery of the EV produce DC power. The synchronous motor of the EV needs AC power to operate, hence an inverter is required. The inverter the Nissan Leaf uses, has an efficiency of 98%, thus the power losses regarding the inverter is deemed negligible [55][56].

3.6 Sensitivity Analysis of Power Output

Figure 3.10 shows the power output of the chosen panel for 21 November 2018 in Stellenbosch. This day was chosen as the day starts of very cloudy and starts to clear up at solar noon. This is desired as it will clearly indicate the relationship between irradiance and output power. The tilt for the panel was set equal to the latitude (33°), as it is generally considered as the optimal tilt angle. The windspeed and ambient temperature for that day are also taken into consideration, and is obtained from weather.sun.ac.za. From Figure 3.10 it is clear to see that the power output is directly proportional to the irradiance, as expected.

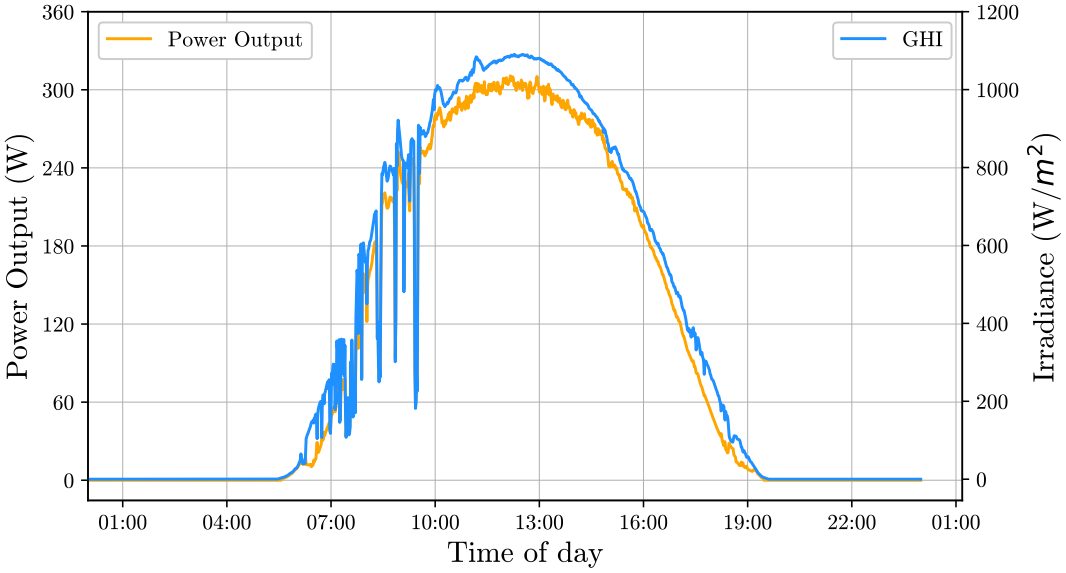


Figure 3.10: Power Output vs GHI.

3.6.1 Effect of Windspeed

To show the effect of windspeed on the power output, all of the other variables were held constant. The STC temperature was used. To clearly show the effect of windspeed, a very clear day was chosen. The day was 3 October 2018 in Stellenbosch. Figure 3.11 shows the relationship between windspeed and output power. Windspeeds of 0 m/s, 17 m/s and 33 m/s were chosen, as it is equal to a car standing still, driving at 60 km/h and driving 120 km/h. It is concluded that windspeed has quite a big effect on the power output and must indeed be considered in the modelling thereof. The reason for the higher windspeed yielding a bigger power output, is that the higher windspeed cools the panel, which in turn makes it more efficient.

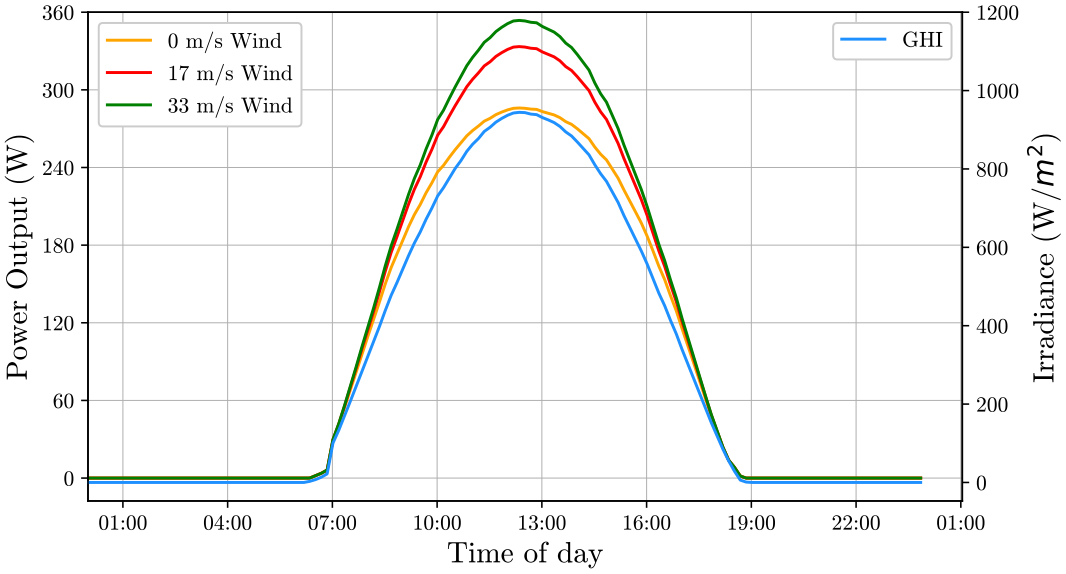


Figure 3.11: Power Output vs GHI at different windspeeds.

3.6.2 Effect of Ambient Temperature

As for showing the effect of windspeed, all the variables will be held constant and only the ambient temperature will be changed. Once again 3 October 2018 was chosen, to keep the comparison accurate. Figure 3.12 shows the relationship between ambient temperature and output power. Three different ambient temperatures were chosen, $5°C$, $25°C$ and $40°C$. These temperatures were chosen as they are consistent with what you could expect when driving around in South-Africa. The windspeed for this investigation was set to 0 m/s, as the windspeed will help cool the panel and this is not what is investigated. The effect of ambient temperature on the power output is quite significant, but not as big as the effect of windspeed. As for the windspeed, a lower ambient temperature will lead the panel to be more effective.

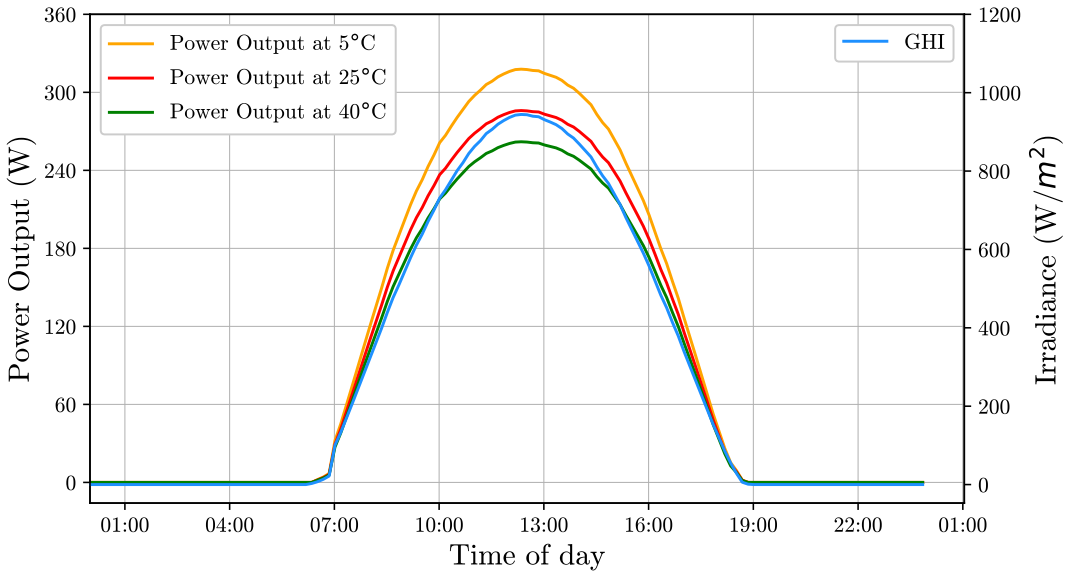


Figure 3.12: Power Output vs GHI at different ambient temperatures.

Chapter 4

GHI Forecasting

4.1 Introduction

GHI forecasting plays a key role in this project. The GHI is needed as it is one of the key variables to determine the power output of the solar panels that is mounted on the roof of the EV. Currently, weather forecasts do not include GHI forecasts, thus it is necessary to do GHI forecasting. The goal for this thesis, in terms of GHI forecasting, is to predict the GHI based only on a clear sky model and cloud cover data from the desired weather station. Further, the forecasting will be done with very little historical data and only using data that is freely available to the public.

The first step to forecasting GHI, is to develop an accurate clear sky model. Once the clear sky model is known, regression techniques will be used to find a correlation between cloud cover, clear sky GHI and actual GHI. The regression will be done based on historical weather and GHI data. This is a very simplistic approach to do GHI forecasting, but it fits the scope of the project very well. The code for the regression techniques can be seen in Appendix 7.6.

4.2 Clear Sky Model

The clear sky model is a model that predicts the GHI for a clear day, i.e. a day with zero clouds (There are various clear sky models discussed in literature, with varying levels of complexity). For this project, the Ineichen and Perez model will be discussed and implemented. This model was chosen due to its relative complexity and it is one of the well known models for determining clear sky GHI. The model was initially developed by using 4 years of measured data. Pvlib provides a function that implements the chosen clearsky model and will be discussed below. The clear sky GHI as described by the Ineichen and Perez model, is defined by the following equation [57]:

$$GHI_{clear} = A \cdot SC \cdot \sin(90 - \theta_z) \cdot exp[-B \cdot \rho_m \cdot (C + D - [T_L - 1])] \quad (4.1)$$

- GHI_{clear} - Clear Sky GHI (W/m^2).
- SC - Solar Constant.
- ρ_m - Optical Air Mass.
- T_L - Linke Turbidy Coefficient.
- A - Altitude Dependant Coefficient.
- B - Altitude Dependant Coefficient.
- C - Altitude Dependant Coefficient.
- D - Altitude Dependant Coefficient.

The solar constant is defined as the rate at which energy from the sun is received on a surface that is perpendicular to the incoming rays, in free space and at the mean distance between the earth and the sun [58]. The value of the solar constant is taken as 1364 W/m^2 for these calculations. The optical airmass can be determined by Equation 4.2 [59].

$$\rho_m = \frac{1}{\cos\theta_z + 0.50572 \cdot [96.07995 - \theta_z]^{-1.6364}} \quad (4.2)$$

The Linke Turbidy coefficient is described as the total optical thickness of a clear sky atmosphere, relative to an aerosol and water damp free atmosphere [30]. The Linke Turbidy coeffecient for a location can be determined by Equation 4.3 [57].

$$T_L = \ln\left(\frac{SC}{DNI}\right) \cdot \left(\frac{9.4 + 0.9 \cdot \rho_m}{\rho_m}\right) \quad (4.3)$$

The Altitude dependant coefficients were calculated using Equations 4.4 through 4.7, where Alt refers to the altitude [57].

$$A = (5.09 \cdot 10^{-5}) \cdot Alt + 0.868 \quad (4.4)$$

$$B = (3.92 \cdot 10^{-5}) \cdot Alt + 0.0387 \quad (4.5)$$

$$C = exp\left(\frac{-Alt}{8000}\right) \quad (4.6)$$

$$D = exp\left(\frac{-Alt}{1250}\right) \tag{4.7}$$

Figure 4.1 shows the predicted clear sky GHI vs the measured GHI. The measured GHI is obtained from www.weather.sun.ac.za and is for 2 February 2019. From Figure 4.1 it is concluded that the chosen clear sky model is sufficiently accurate for the purposes of this project.

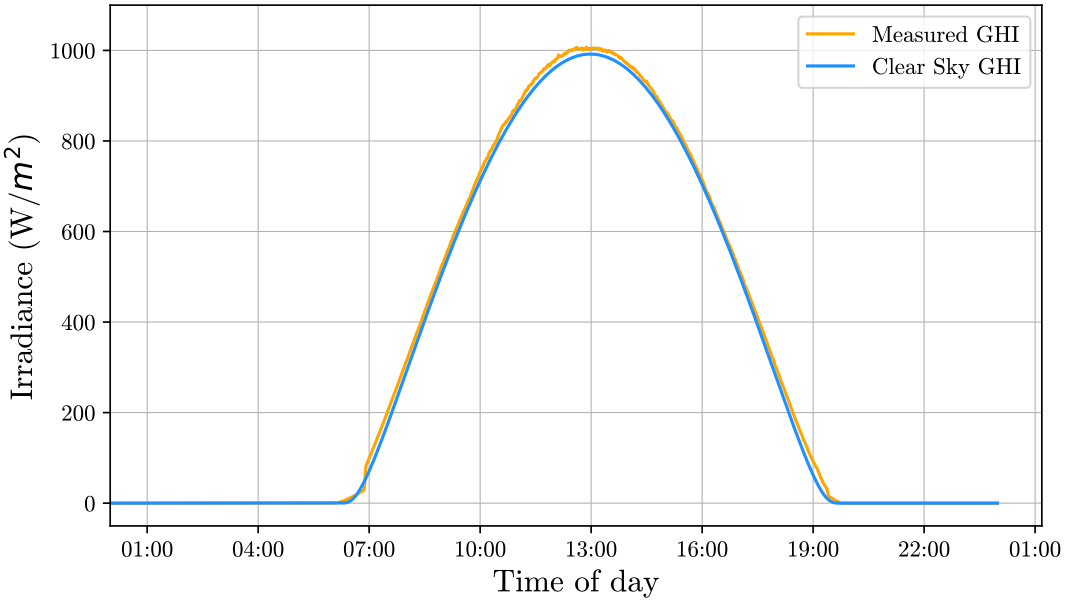


Figure 4.1: Clear Sky GHI Validation.

4.3 Regression Techniques

To find the relationship between cloud cover, clear sky GHI and actual GHI, regression techniques are used. There are a few regression techniques that will be discussed and the most accurate one will be chosen. Regression is defined as a predictive modelling technique used to define the relationship between a dependant variable and independent variable(s) [60]. In short, regression takes the training dataset (both dependant and independent variable(s)) and uses it to draw a line of best fit. This line of best fit is then used to predict the dependant variable from the given independent variable(s).

Python provides a built in library for implementing the various regression techniques. The different regression techniques will be discussed in short, as the scope of this project does not entail a deep understanding of regression techniques. Larson et al. discusses a model for determining GHI from cloud cover

and clear sky data [61]. This will be used as a benchmark for the regression techniques discussed in this chapter. The model for determining the predicted GHI by Larson et al. is described by Equation 4.8, were CC is the cloud cover (0 - 1) [61].

$$GHI = GHI_{clear} \cdot [0.35 + 0.65 \cdot (1 - CC)] \tag{4.8}$$

The cloud cover is received from Yr.no and is expressed as a percentage between 0%- 100%. Yr.no allows you to make API calls to retrieve the latest weather forecast, and this forecast is updated every 12 hours. For the forecasting done in this project, there will be distinguished between low-, medium- and high clouds. The reason for distinguishing between the different cloud heights, is that they absorb and reflect radiation at different rates. Lower clouds tend to reflect radiation back to space, where higher clouds reflect the radiation towards the surface of the earth [62]. The historic GHI measurements are obtained from www.weather.sun.ac.za. The regression models will be trained with only 16 days worth of historical data.

4.3.1 Linear Regression

Linear regression finds the relationship between the variables by making use of a linear line of best fit. In this case, the line of best fit is a straight line. The equation used to do linear regression is defined by Equation 4.9 [60]. Figure 4.2 show the Linear Regression line for a randomly generated dataset. The intercept and slope are determined by making use of the least square method, and is discussed by Equations 4.10 and 4.11 [63]. The least square method is used to minimize the error between the line of best fit and the given dataset [63]. Figure 4.3 shows the predicted GHI by making use of linear regression.

$$Y = \beta_0 + \beta_1 X \tag{4.9}$$

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \tilde{x})(y_i - \tilde{y})}{\sum_{i=1}^{n}(x_i - \tilde{x})^2} \tag{4.10}$$

$$\beta_0 = \tilde{y} - \beta_1 \tilde{x} \tag{4.11}$$

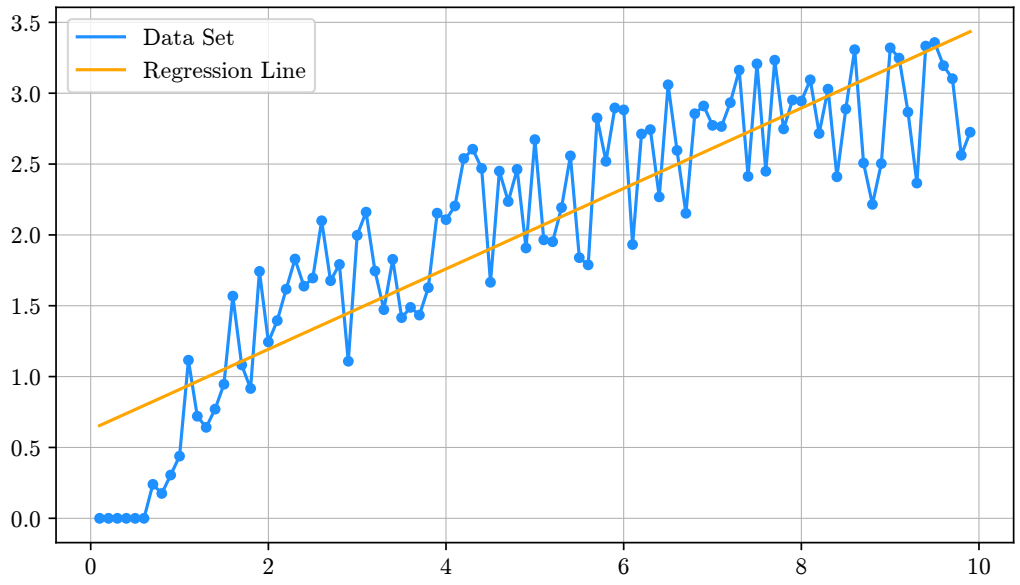


Figure 4.2: Linear Regression Line.

- Y - Dependant Variable.
- X - Independent Variable.
- β_0 - Intercept.
- β_1 - Slope.
- $\tilde{y}$ - Average of Y in Dataset.
- $\tilde{x}$ - Average of X in Dataset.

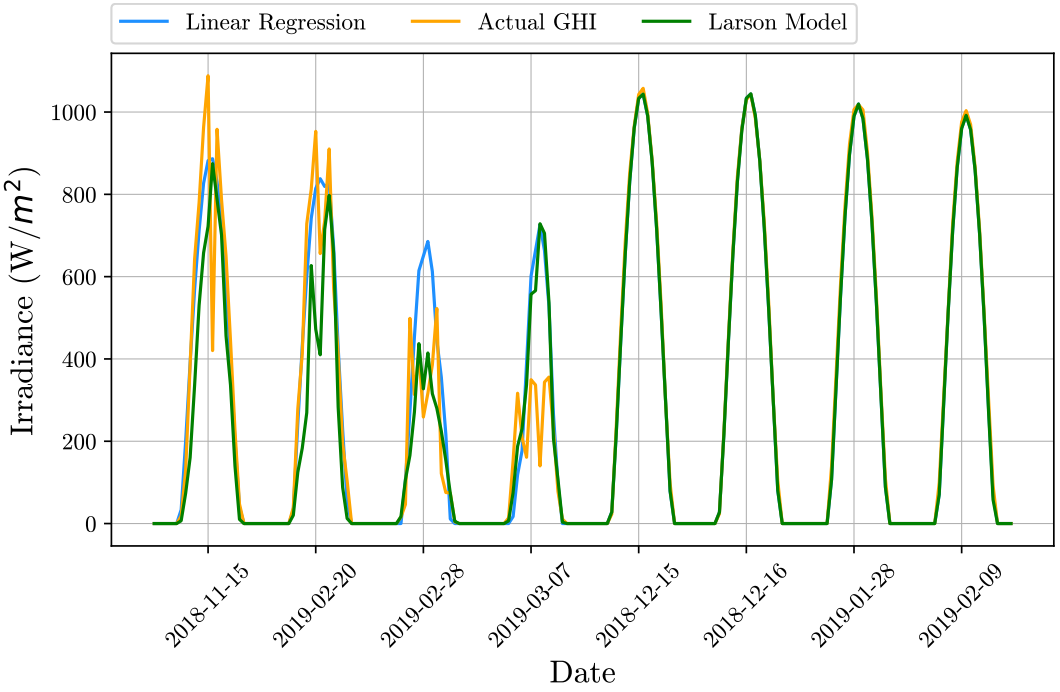


Figure 4.3: Linear Regression Model.

4.3.2 Polynomial Regression

Similar to linear regression, polynomial regression finds a line of best fit. The difference is, that with polynomial regression, the line of best fit is a polynomial line. Figure 4.4 shows the Polynomial Regression line of the 6th degree for a randomly generated dataset. The equation used to determine the polynomial regression line, is defined by Equation 4.12 [64]. In order to compute the regression line, Equations 4.13 through 4.15 have to be implemented. This is needed in order to minimize the error by making use of the least square method [65]. When using polynomial regression, it is import to find the correct order. To high of an order will lead to over-fitting, where to low of an order will lead to under-fitting. Figure 4.5 shows the predicted GHI by making use of polynomial regression. Different polynomial degrees were compared, and it was found that the second degree produced the most accurate prediction for this dataset.

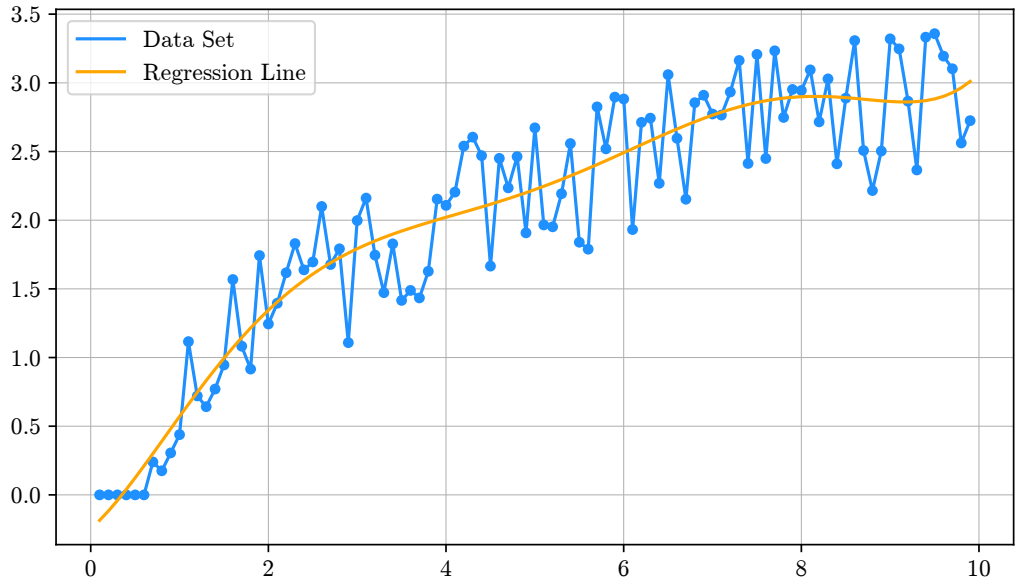


Figure 4.4: Polynomial Regression Line.

$$Y = \beta_0 + \beta_1 X + \beta_1 X^2 + ... + \beta_n X^n \tag{4.12}$$

$$\bar{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ Y_m \end{bmatrix} \tag{4.13}$$

$$\bar{X} = \begin{bmatrix} X_1^0 & X_1^1 & X_1^2 & \dots & X_1^n \\ X_2^0 & X_2^1 & X_2^2 & \dots & X_2^n \\ \dots & \dots & \dots & \dots & \dots \\ X_m^0 & X_m^1 & X_m^2 & \dots & X_m^n \end{bmatrix} \tag{4.14}$$

$$\bar{\beta} = (\bar{X}^T \cdot \bar{X})^{-1} \cdot (\bar{X}^T \cdot \bar{Y}) \tag{4.15}$$

- β - Regression Coefficients.
- $\bar{Y}$ - Dependant Variable Matrix.
- $\bar{X}$ - Independent Variable(s) Matrix.
- $\bar{\beta}$ - Regression Coefficient Matrix.

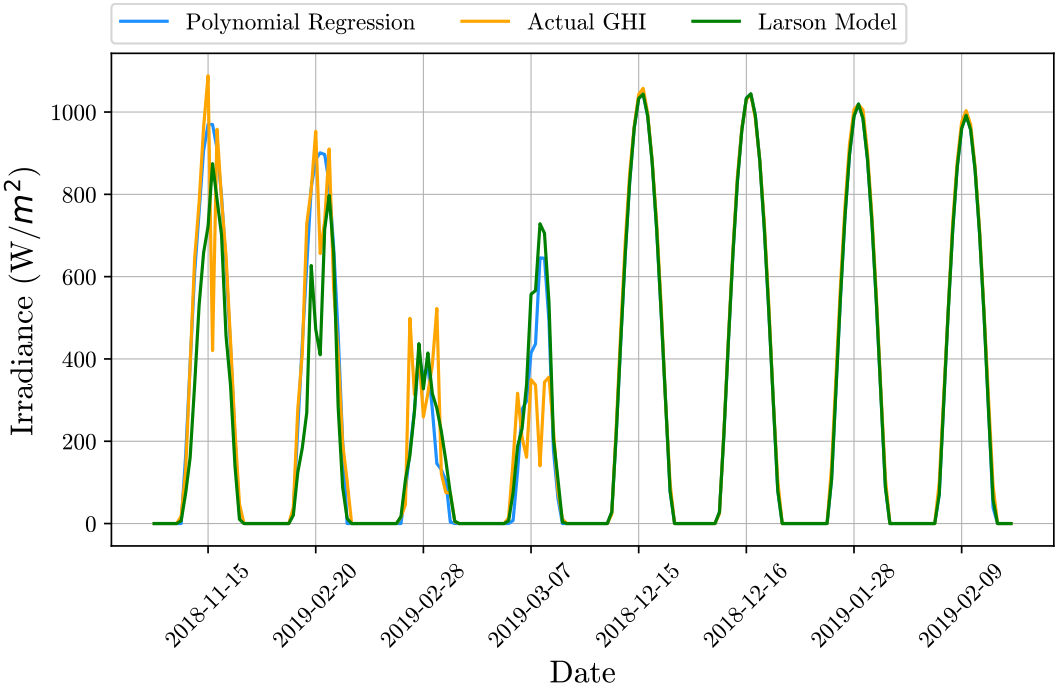


Figure 4.5: Polynomial Regression Model.

4.3.3 Logistic Regression

Logistic regression is generally used for classification problems, but for this application it will be used for prediction. Figure 4.6 shows the logistic regression line for a randomly generated dataset. To use logistic regression for prediction, the dependant variables from the training data have to be divided into classes [66]. This is done by making use of the Scikit-learn library from Python, which transforms the data into classes and inverses the transform after the calculations are done [67].

A big difference from the previous regression techniques, is that logistic regression determines the probability of each dependant-variable-class based on the independent variable(s) and the class with the highest probability is the predicted values (once it is inverse transformed as mentioned previously). The probability of each of the so called dependant-variable-classes is calculated by Equation 4.16, where p is the probability [66]. The regression coefficients are determined by making use of the maximum likelihood method, and is explained by Equation 4.17, where l is the likelihood [68]. To do so, the equation is differentiated, set equal to zero and then solved. Figure 4.7 shows the predicted GHI by making use of logistic regression.

$$p(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}} \tag{4.16}$$

$$l(\beta_0, \beta_1) = \sum_{i=1}^{n} -\log 1 + e^{-(\beta_0+\beta_1 X_i)} + \sum_{i=1}^{n} Y_i(\beta_0 + \beta_1 X_i) \tag{4.17}$$

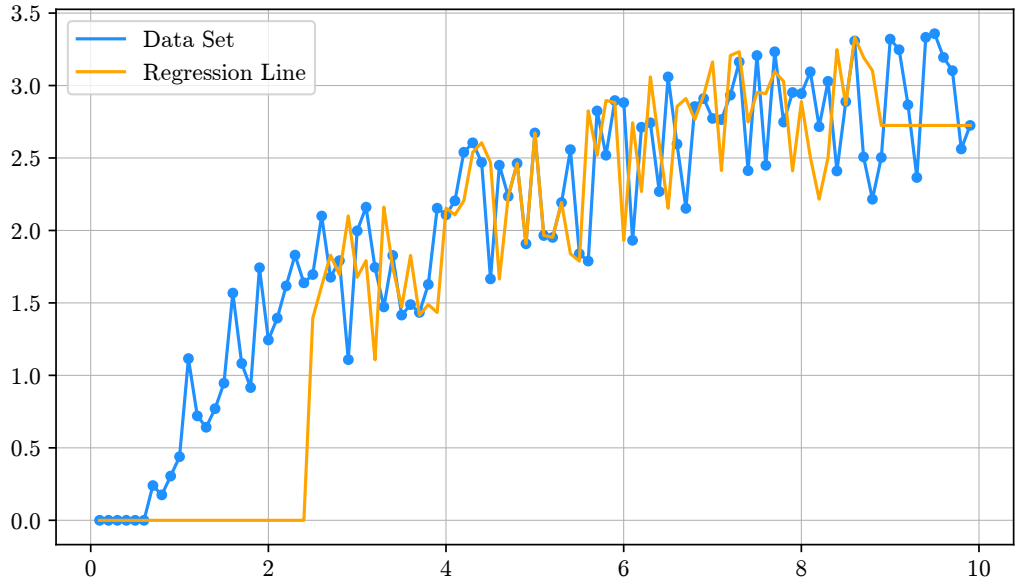


Figure 4.6: Logistic Regression Line.

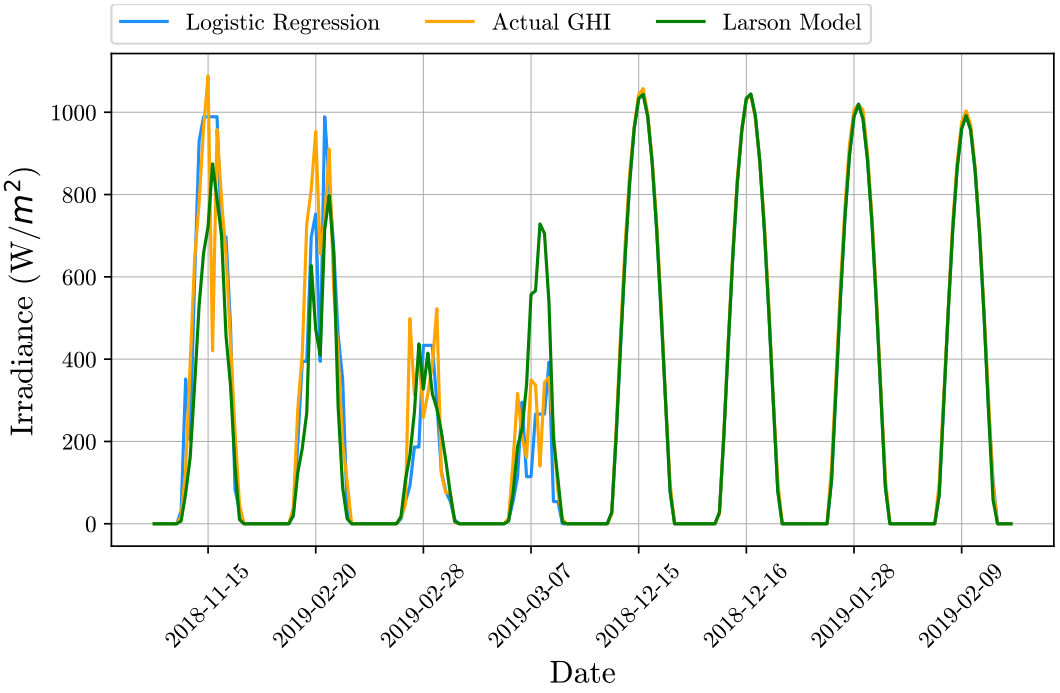


Figure 4.7: Logistic Regression Model.

4.4 Results

To review the accuracy of the prediction models, four Key Performance Indicators (KPI) will be used. They are Mean Absolute Error (MAE), Mean Bias Error (MBE), Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE). When implementing these KPIs, only the instances between sunrise and sunset will be taken into consideration. The points in the night were the GHI is 0, will be ignored. This is done, because roughly half of a day there will be no GHI, and this will cause the KPIs to be skewed.

MAE is the average of the absolute of the difference between the predicted and actual values. For this application, the MAE will state how many Watt per meter squared the prediction will be out on average every hour. This is a good indicator, as it is a quantifiable result. The MAE can be determined by implementing Equation 4.18, where n is the number of data-points (keeping in mind that for this application it will only include the predicted GHI from sunrise to sunset) [69].

$$MAE = \frac{1}{n} \cdot \sum_{n} |GHI_{measured} - GHI_{predicted}| \tag{4.18}$$

MBE is the average of the difference between the predicted and actual values. MBE is a good KPI, in the sense that it will indicate whether the model is over- or under-predicting. A positve MBE will indicate under-prediction and a negative MBE will indicate over-prediction [70]. The MBE can be determined by implementing Equation 4.19 [70].

$$MBE = \frac{1}{n} \cdot \sum_{n} (GHI_{measured} - GHI_{predicted}) \tag{4.19}$$

MAPE is the average of the percentage that each predicted value differs from the corresponding actual value. This KPI is not considered accurate, as it is skewed, because if there is a big difference between the predicted and actual value for a low actual value, this will have a very big impact on the MAPE. The same error for a higher actual value, will have much less of a impact on the MAPE [69]. The MAPE can be determined by implementing Equation 4.20 [69].

$$MAPE = \frac{1}{n} \cdot \sum_{n} \frac{|GHI_{measured} - GHI_{predicted}|}{GHI_{measured}} \cdot 100 \tag{4.20}$$

RMSE is defined as the square root of the average error squared. RMSE is a KPI that gives big attention to the magnitude of the error. If big differences between predicted and actual vales are very undesirable, RMSE will be the perfect KPI. The RMSE can be determined by implementing Equation 4.21 [69].

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_{n} (GHI_{measured} - GHI_{predicted})^2} \tag{4.21}$$

Table 4.1 shows the normalised Key Performance Indicators for the various prediction models, as well as for the discussed clear sky model. To normalise the MAE, MBE and RMSE, they are simply divided by the average of $GHI_{measured}$ [70]. The reason for normalising the KPI, is to make it comparable to models that predict something different or simply uses other data than is used in this project. The model that did the best in the the respective KPIs is printed in blue.

From Table 4.1 it is clear that the Polynomial regression model is the most accurate. It has the lowest MAE, lowest MBE and the lowest RMSE. It is important to note that the polynomial regression model slightly over-predicts and to take this into consideration when using the predicted GHI to calculate power yield. The clear sky model is also reviewed and is very accurate. The clear sky model slightly under-predicts, which is desirable when doing power-yield predictions.

Table 4.1: Prediction Accuracy

	nMAE	**nMBE**	**MAPE**	**nRMSE**
Larson Model	40.49%	12.95%	52.79%	54.73%
Linear Regression	33.83%	-12.79%	59.15%	47.53%
Polynomial Regression	**24.34%**	**-2.86%**	44.22%	**39.71%**
Logistic Regression	29.77%	7.29%	**41.51%**	41.88%
Clear Sky Model	2.77%	2.72%	11.4%	2.99%

Chapter 5

Software Development

5.1 Introduction

At this point in the project, all of the theory is explained and completed. However, everything still has to be integrated and implemented. To do this, software was developed in the Python environment. Python was chosen as it is regarded as a high level programming language, which has a good ease of use. Python also has a large and comprehensive built-in standard library. As this project is strictly theoretical, this section will play a big role in the justification of the project. The core parts of the code discussed in this section can be seen in Appendix 7.6.

5.2 Road Profile Data

The road profile data is a very important part of this project. This data is needed in order to calculate the power requirements for a specific route and thus to compute the range of the EV. The power contribution from the solar panels on the roof of the EV, is highly dependant on the road profile data. As the EV drives, the tilt angle and surface Azimuth of the PV panels will be changing constantly, and in turn this will effect the total in-plane irradiance. If the road profile data is incorrect or inaccurate, all of the modelling making use of it will consequently also be inaccurate. The Latitude and Longitude for each point along the desired/chosen route is retrieved from Google Earth. Sadly, Google Earth does not provide all the information needed to complete the road profile data. To determine the rest of the needed information, mathematical calculations will be needed, and will be discussed below. An example of the road incline and -elevation versus distance can be seen in Figure 5.1, for a route from Stellenbosch to Robertson.

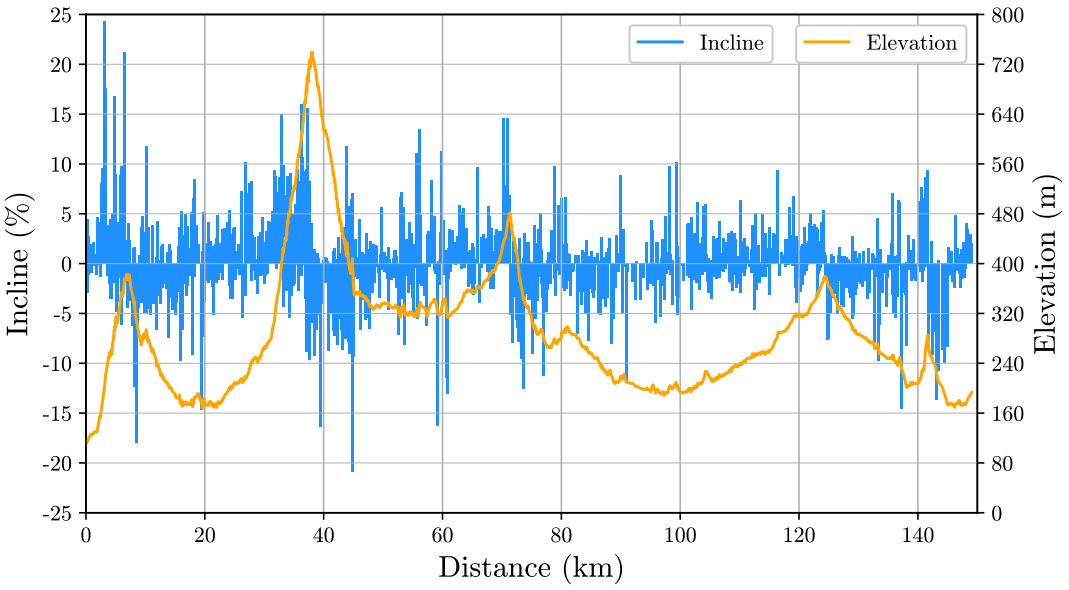


Figure 5.1: Road Incline vs Elevation.

5.2.1 Distance Between points

The distance between two Latitude and Longitude points can be determined by making use of the Haversine formula. This formula is slightly flawed in the sense that it takes the shortest distance between the two points. Thus the accuracy of Haversine formula in this application, will be dependant on the frequency of coordinate-points provided by Google Earth. This will however not be a problem when calculating the range of the EV, as the model uses the distance provided by the road profile data and not the actual distance. The Haversine formula is broken up to make it easier to understand and can be seen in Equations 5.1 through 5.3.[71]

$$A_{hav} = \sin^2\left(\frac{L_2 - L_1}{2}\right) + \cos(L_1) \cdot \cos(L_2) \cdot \sin^2\left(\frac{Long_2 - Long_1}{2}\right) \tag{5.1}$$

$$B_{hav} = 2 \cdot \arctan\left(\frac{\sqrt{A_{hav}}}{\sqrt{(1 - A_{hav})}}\right) \tag{5.2}$$

$$Dist = R_{earth} \cdot B_{hav} \tag{5.3}$$

- A_{hav} - Constant.
- B_{hav} - Constant.
- $L_{1,2}$ - Latitude Points ($^\circ N$).
- $Long_{1,2}$ - Longitude Points ($^\circ E$).

- $Dist$ - Distance Between Points (m).
- R_{Earth} - Earth Mean Radius (m).

A function is written using Python, which uses a for loop (to step through every coordinate point retrieved from Google Earth) to implement Equations 5.1 through 5.3, to calculate the distance between each of the Latitude and Longitude points retrieved from Google Earth. These distance points are placed into a data-frame within the for loop and is then merged with the data-frame containing the Latitude and Longitude points.

5.2.2 Heading

The Heading is equal to the surface Azimuth angle, and is required to determine the angle of incidence and in turn the total in-plane irradiance, as mentioned in Chapter 3.2. The heading from point 1 to point 2 can be determined by Equations 5.4 through 5.6 [72].

$$Head = \arctan\left(\frac{A_{Head}}{B_{Head}}\right) \tag{5.4}$$

$$A_{Head} = \cos(L_2) \cdot \sin(Long_2 - Long_1) \tag{5.5}$$

$$B_{Head} = \cos(L_1) \cdot \sin(L_2) - \sin(L_1) \cdot \cos(L_2) \cdot \cos(Long_2 - Long_1) \tag{5.6}$$

- $Head$ - Heading $(°)$.
- A_{Head} - Constant.
- B_{Head} - Constant.

As for the distance, a function is written using Python, which uses a for loop to determine the heading between each of the Latitude and Longitude points extracted from Google Earth. These headings are placed into a data-frame within the for loop and is then merged with the data-frame containing the Latitude, Longitude and distance points.

5.2.3 Road Angle

In order to determine the angle of the road, it is necessary to know the elevation/altitude of each latitude-longitude point. With the elevation of each point in the road profile data known, as well as the distance between the points, the road angle can be determined by implementing simple trigonometry. The road angle is determined by Equation 5.7.

$$\theta_{Road} = \arctan\left(\frac{\Delta Alt}{Dist}\right) \tag{5.7}$$

- θ_{Road} - Road Angle (rad).
- Alt - Altitude or Elevation (m).
- ΔAlt - Change in Elevation (m).

The Google Maps Platform provides an elevation Application Programming Interface (API). This API allows the user to get the elevation (in meters) for any location, given the latitude and longitude thereof. Python code is written to make an API request for each point in the road profile data, to find the elevation of each point. Once the elevation is known, the road angle is determined by implementing Equation 5.7 in a for loop, and adding it to the before mentioned data-frame to fully complete the road profile data.

5.3 Speed Instructions

To determine the power that is required to drive at a certain speed at a certain incline, Equation 2.1 is used. One key variable needed to determine the power requirements is the acceleration of the EV. Acceleration can be determined by making use of Equation 5.8 and 5.9. The modelled EV can accelerate from 0 to 60 mph in roughly 10 s, which equates to a maximum acceleration of 2.68 m/s^2 [5].

$$a_{EV} = \frac{\Delta v}{T_{EV}} = \left(\frac{2Dist}{T_{EV}^2} - \frac{2v_i}{T_{EV}}\right) \tag{5.8}$$

$$T_{EV} = \frac{Dist}{v} \tag{5.9}$$

- v_i - EV Initial Speed (m/s).
- Δv - Change in EV speed (m/s).
- T_{EV} - Time (s).

With the road profile being known and the total tractive effort and acceleration computable, the power requirements for a route is calculated as follows: As mentioned in Chapter 2.2, the Nissan Leaf has a 80 kW motor, thus this is the limit for the available power of the EV. Python is used to develop an algorithm that calculates the maximum speed that the EV can drive at a certain road incline with a certain power limitation (80 kW for now).

The algorithm populates a data-frame with speeds incrementing from 0 to 120 km/h in small steps. The power required to drive each of these speeds at the specific road incline is calculated by implementing Equation 2.1 to 2.8 and Equation 5.8 and 5.9. This method is repeated to create a new virtual lookup table for each point in the road profile data. An example of this virtual lookup

table, but for various road inclines, can be seen in Figure 5.2. From Figure 5.2 it can be seen that the fastest the EV can drive on a road with a 20% incline, is about 85 km/h. With the user defined speed limit taken into consideration, a data-frame is populated with the maximum allowable driving speed for each point in the road profile data, without exceeding the speed limitation and the power limitation of 80 kW. For this project, this data-frame is referred to as the speed-instruction data-frame.

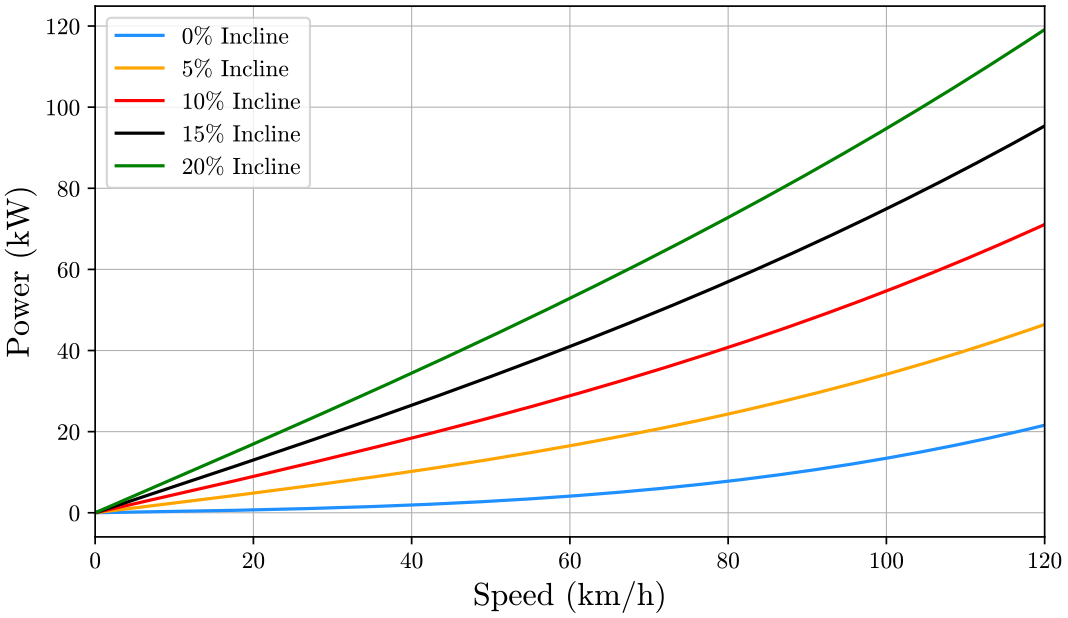


Figure 5.2: Speed vs Power Required.

At this point, the theoretical maximum driving speed for each point along the route is known. As previously mentioned, the EV has a maximum acceleration of 2.68 m/s^2, and this has to be taken into consideration. The maximum acceleration of the EV will determine whether the EV will be able to reach the speed that is in the speed-instruction data-frame, within the distance at hand. With the distance between the points in the road profile data known, the maximum speed that the EV will be able to accelerate to, from the previous speed in the speed-instruction data-frame is calculated. If the speed in the speed-instruction data-frame is unattainable, it is replaced with the maximum speed the EV will be able to reach within that specific distance, with a maximum acceleration of 2.68 m/s^2. To summarise, at this point the algorithm determines the speed-instruction data-frame, which contains the maximum speed the EV is able to attain for a route, without requiring more than 80 kW or exceeding an acceleration of 2.68 m/s^2 and the defined speed limit. The algorithm also returns that actual power that is used in order to maintain or reach the corresponding speed. The power used will not always be equal to the power limitation, as the speed limit and the road incline will have an effect. For

example, when driving down a steep incline, not much power will be needed to reach the speed limit.

5.3.1 Power Limitation

The algorithm is now altered in such a way that the power limitation can be changed anywhere between 0 and 80 kW. The main reason for this, is to lower the average speed within the speed-instruction data-frame. Adjusting the speed limitation, will only ensure that the speed limit is never exceeded, but adjusting the power limitation will cause the maximum achievable speed for each point in the road profile data to be lower.

Figure 5.3 shows the speed versus distance for a route with inclines decreasing from 20% to 0%, in 5% increments every 20 km. Firstly the power limitation was set to 40 kW with a speed limit of 120 km/h, to show how the EV will be able to drive faster as the road incline lowers and act as a baseline. Next, the power limitation was set to 20 kW with the speed limit kept at 120 km/h. Lastly, the power limitation was kept at 40 kW, but the speed limit was lowered to 60 km/h. From Figure 5.3, it can be seen that there is clear difference between applying a speed limit versus applying a power limitation. With the speed limit being halved, the curve will follow the baseline speed exactly, up until the speed limit is reached. But with the power limitation being halved, the curve never reaches the baseline speed. This power limitation will later be used to control the energy required to complete a certain route.

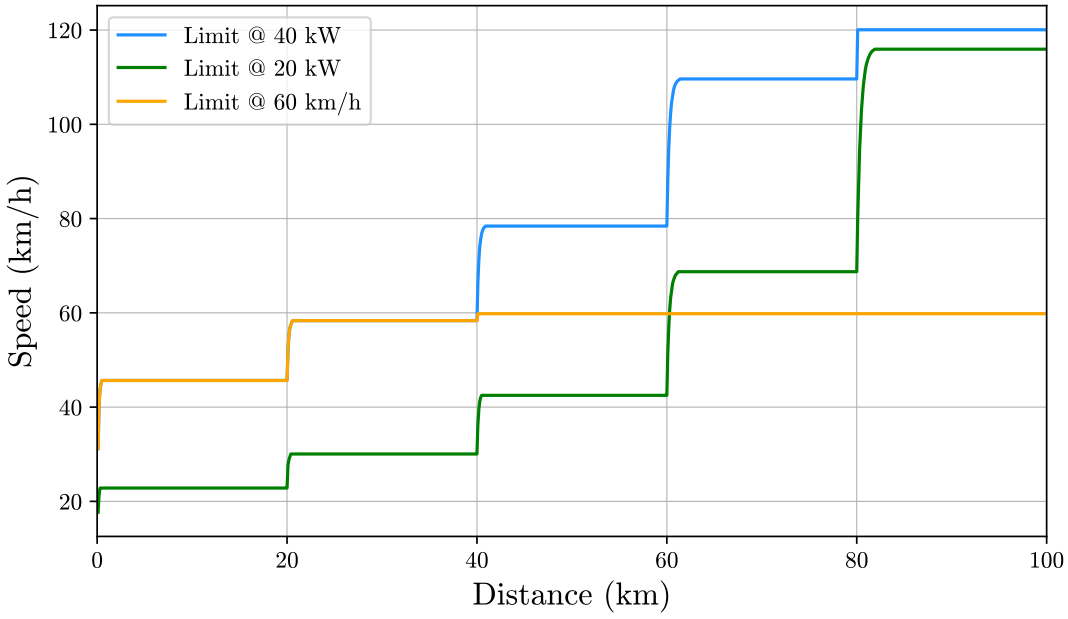


Figure 5.3: Power- vs Speed Limitation.

5.3.2 Adaptive Speed Instructions

Currently, the speed-instruction algorithm provides the driver with a target speed to follow for each point along the road, which will correspond with a certain range. The user may however not always reach the target speed due to traffic jams, accidents or obstructions in the road. A new function was added to the algorithm, which takes the actual speed the driver obtained into account, when determining the next reachable/target speed. This functionality is added to make the algorithm applicable to actual driving conditions.

Figure 5.4 shows the speed instructions and adaptive speed instructions, for the route in Figure 5.1, with a speed limit of 120 km/h and a power limitation of 80 kW. Figure 5.5 shows the speed instructions and adaptive speed instructions for a section of the route, to make the change in speeds more visible. The actual driving speed the driver achieved was set to 30 km/h, as the scenario may be that the driver can be stuck in a road segment between road works. The adaptive speed instructions are calculated by taking into account that the driver did not reach the speed of the original speed instructions, but only reached 30 km/h. The adaptive speed instructions give the new target speed based on a previously achieved speed of 30 km/h.

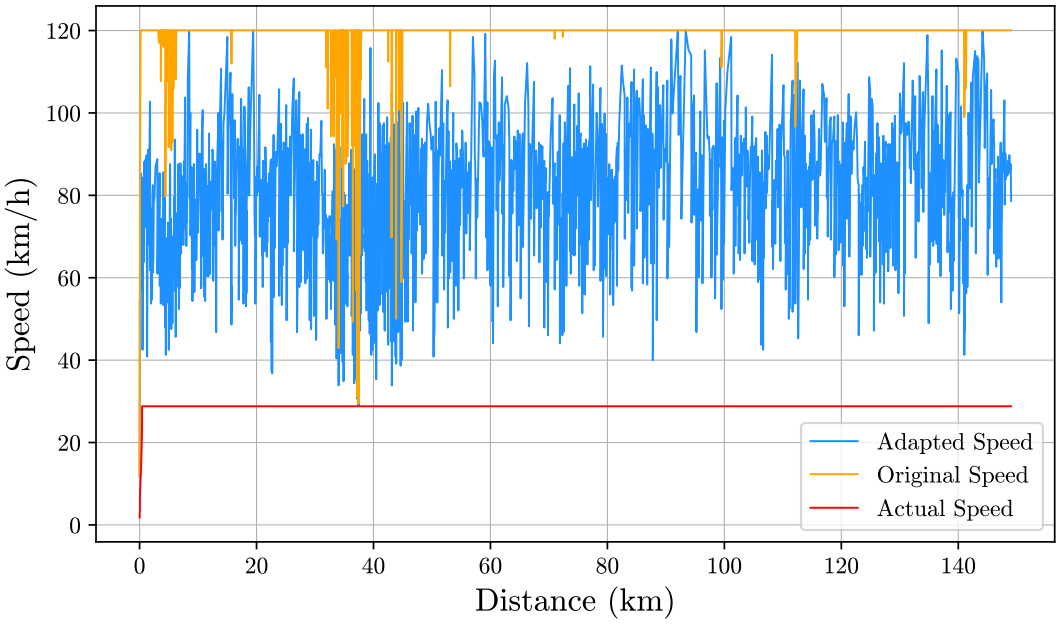


Figure 5.4: Adaptive Speed Instructions.

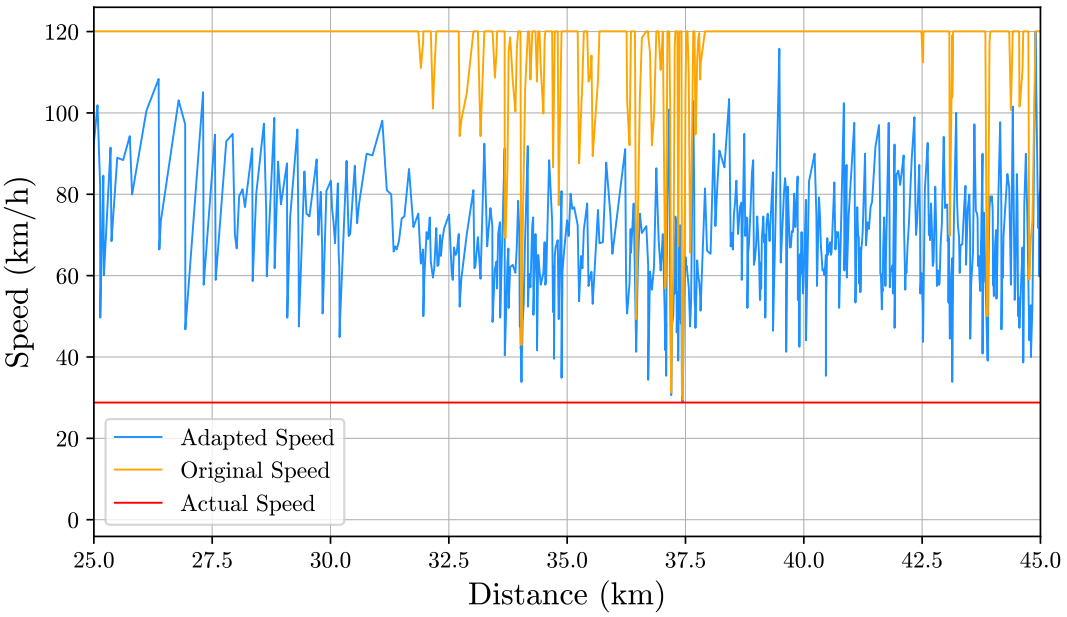


Figure 5.5: Adaptive Speed Instructions.

5.4 Energy Requirements

Up to this point, the maximum achievable speed for a specific route can be determined. This can be done with or without a speed- or power limitation defined. In order for the algorithm to be applicable to an EV, the energy requirements have to be calculated. The energy requirements have to be calculated as the EV makes use of a battery as the energy source. It is important to take note of the difference between power and energy. Power is the rate at which energy is used and is measured in Watt (W). Energy is the amount of power that is used in a specific time and is measured in Watt per hour (Wh).[73] Energy is determined by the product of power and time. For example, if a light-bulb uses 100 W and is burning for 10 hours straight, the light-bulb would have used 1 kWh of energy. The Nissan Leaf has a 30 kWh battery, which means that it can produce 30 kW for 1 hour, 1 kW for 30 hours or any ratio of power and time that equates to 30 kWh.

A new algorithm was developed that determines the battery capacity as the vehicle drives. With the speed and power provided by the speed-instruction algorithm, the driving time between points can be determined. The energy needed to drive at the instructed speed, for the calculated time, is determined by the product of the power and the time. The battery capacity is then updated, by subtracting the energy needed at every point in the road profile data, from the available battery capacity. This battery algorithm takes the speed, power and battery capacity as inputs, and returns the available battery

capacity at each point of the route, as can be seen in Figure 5.6.

Figure 5.6 shows the battery capacity vs distance for the same route as in Figure 5.1, with the same speed- and power limitations as in Figure 5.3 with the addition of a 80 kW and 120 km/h limit. For the case with the 80 kW power limitation and 120 km/h speed limit, it is clear to see that the EV will run out of energy before the destination is reached. For the other 3 cases, the EV will have sufficient energy to complete the route, with the 40 kW power- and 60 km/h limitation being the most energy efficient. Figures 5.7 and 5.8 shows the power and speed versus time for the example above, for the same fraction of the route as in Figure 5.5. The duration of the trip for the various limitations can be seen in Table 5.1.

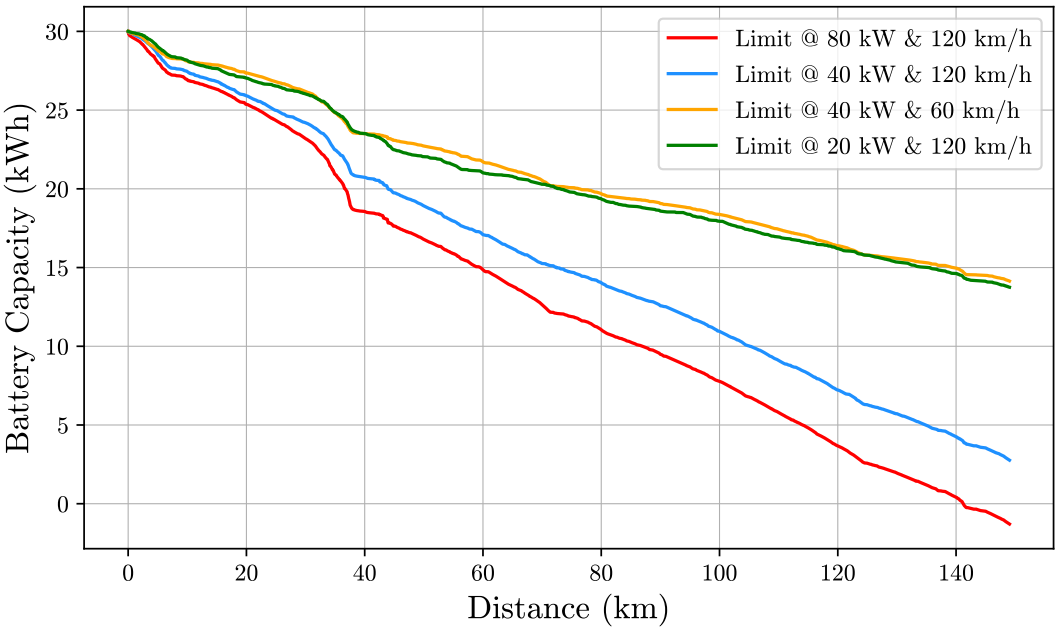


Figure 5.6: Battery Capacity vs Distance.

Table 5.1: Time Travelled

Limits	**Time Travelled**
80 kW and 120 km/h	1h 15min
40 kW and 120 km/h	1h 20min
40 kW and 60 km/h	2h 10min
20 kW and 120 km/h	1h 45min

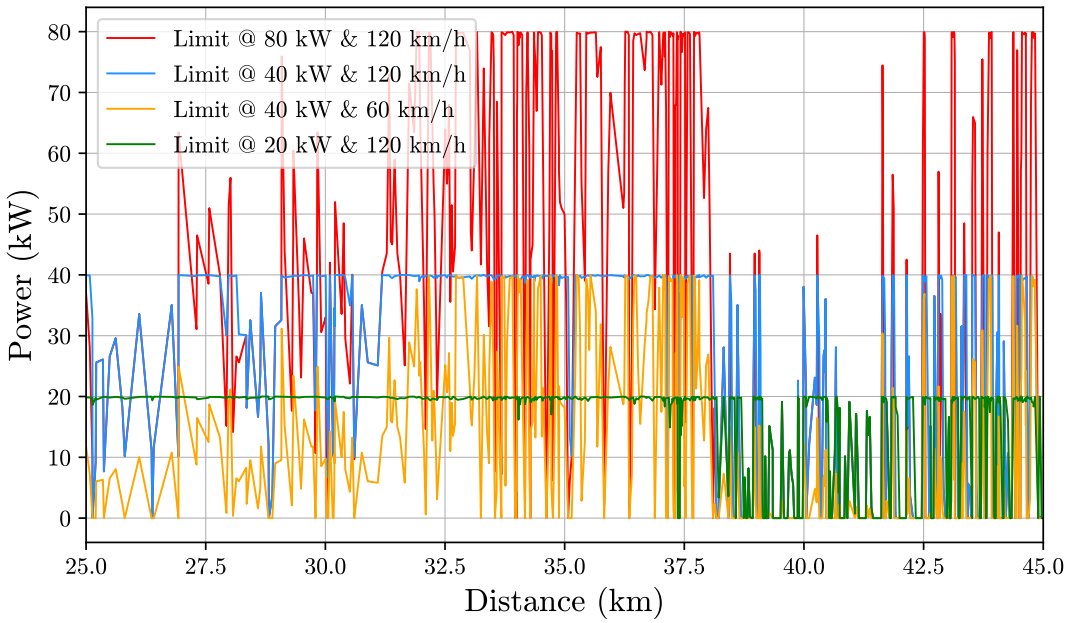


Figure 5.7: Power Usage vs Distance.

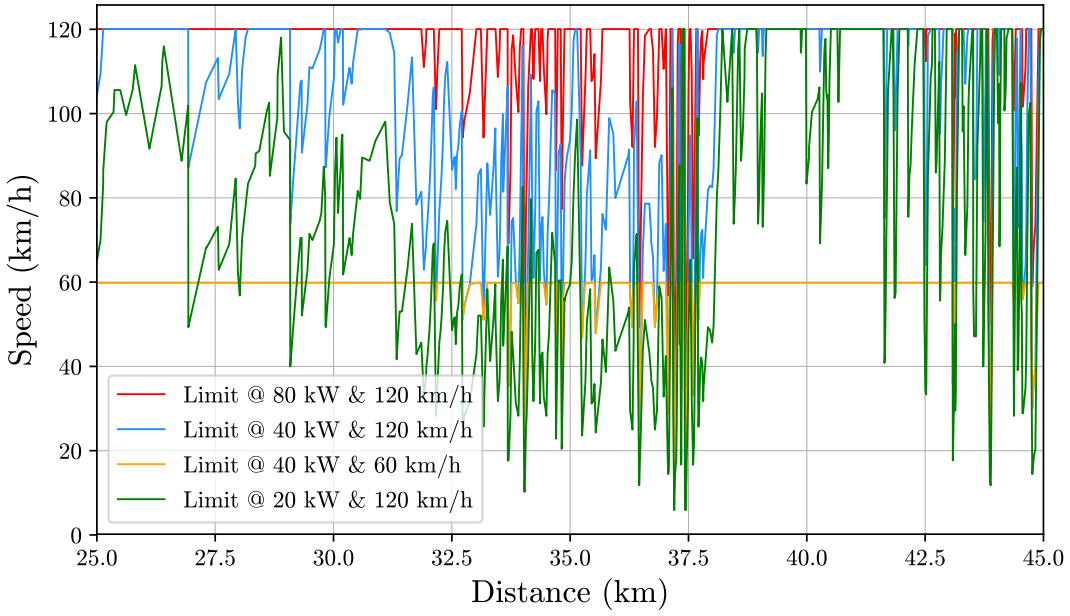


Figure 5.8: Speed vs Distance.

5.5 Solar Contribution

It is decided that two PV panels will be mounted on the EV, which can be on the roof, bonnet or boot. As mentioned in Section 3, all of the losses regarding PV yield will be neglected.

5.5.1 Driving Contribution

To determine the contribution from the solar panels, the power output is determined for every point in the road profile data. To do this, the total in-plane irradiance has to be determined for each point in the road profile data, as mentioned in Chapter 3. The windspeed the panels experience, is set equal to the velocity of the EV. For this project, external wind is neglected, as it is assumed that the wind from the EV driving, will be dominant. The tilt- and surface Azimuth angle is contained within the road profile data, with the tilt being equal to the road angle and the surface Azimuth being equal to the heading.

With the power contribution from two PV panels known, the power from the panel is subtracted from the power required for the corresponding point in the road profile data. This will update the required power, and in turn will affect the energy requirements of the EV. In other words, while the EV is driving, the power from the PV will not be used to charge the battery, but rather directly go to the motor. This will lead to less power being drawn from the battery. Figure 5.9 shows the battery capacity vs distance for the same limitations and route as in Figure 5.6, but with the contribution from the PV panels included. The GHI was set to 1200 W/m^2 for the whole route. GHI of 1200 W/m^2 is a bit unrealistically high, but it is chosen as it will more clearly show what affect the PV panels can have on the range of the EV and it is only used for comparative purposes. It can be seen that the case with the smallest speed limit, will lead to the biggest save in battery capacity.

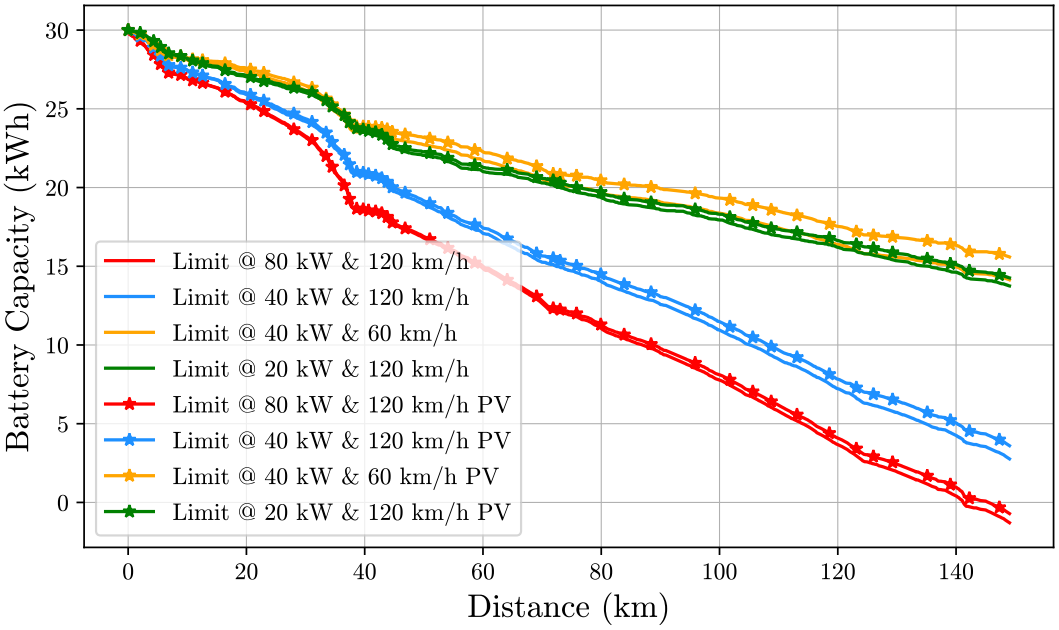


Figure 5.9: Power Usage vs Distance.

5.5.2 Charging Contribution

For the scenario where the driver of the EV has a job from 08:AM to 17:00PM, the battery of the EV will be able to charge while the driver is at work. To determine how much the battery will charge, the charging profile of the Nissan Leaf has to be investigated. The charge controller of the Nissan Leaf has a power limitation of 6.6 kW. This is ideal, as the power produced by the two PV panels will never exceed 6.6 kW. To determine how much the battery will be charged, Equation 5.10 is used. Equation 5.10 is a very simplified approach to determine the charge of the battery, but it is accurate in this case, as the power limitation of the charge controller can never be exceeded.

$$SOC = \frac{P_{PV} \cdot T_{PV}}{B_{cap}} \cdot 100 \tag{5.10}$$

- SOC - State Of Charge (%).
- P_{PV} - Power Produced by PV Panels (kW).
- T_{PV} - Time (s).
- B_{cap} - Battery Capacity (kWh).

In a theoretical scenario, where the EV receives 1000 W/m^2 of irradiance for 9 hours, each PV panel would produce 350 W and the battery would get charged 21%. Figure 5.10 shows the level of charge the battery can reach if parked for one hour at different irradiance levels. It is assumed that the EV

is not parked on a incline, there is no wind and the ambient temperature is $25°C$. As expected, higher GHI will lead to a higher charge capacity, as the PV panels produce more power. Results regarding the amount of charge gained with a normal clear- and overcast day, will be discussed in Chapter 6.

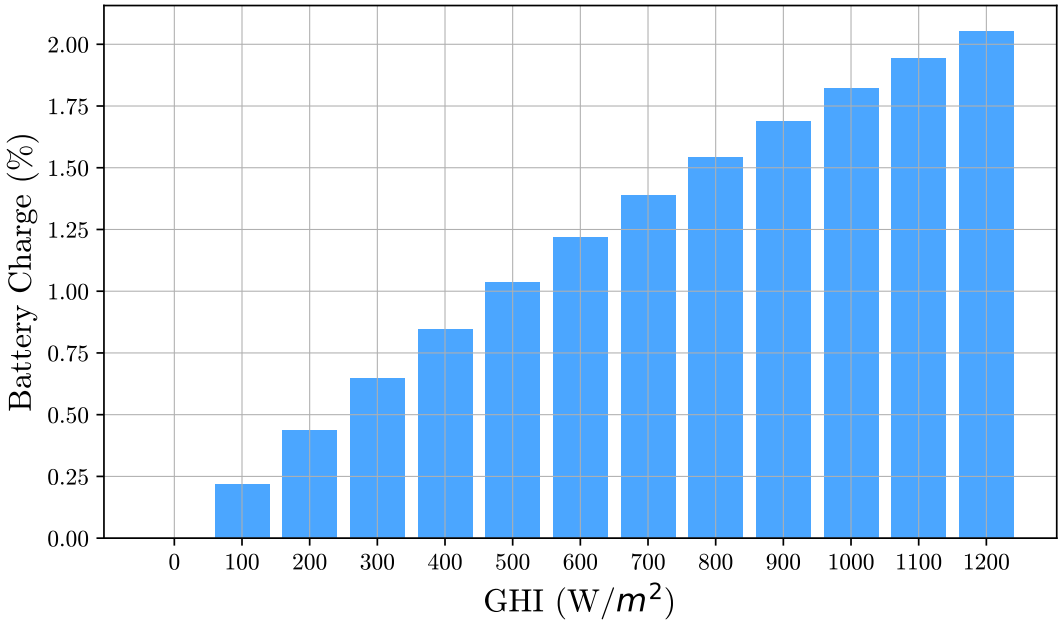


Figure 5.10: Battery Charge vs GHI.

Chapter 6

Results

6.1 Graphical User Interface

As a way to help the driver of the EV understand and use the algorithms developed in this project, a GUI has been developed. The GUI was developed by making use of Pythons' graphical interface, Tkinter. Tkinter allows you to link buttons with functions. For functions that require input from the user, a so called Entry widget is used to obtain information from the user. The placement of the buttons, text, plots, etc. is done by creating different frames. These frames work like a grid that contains rows and columns. For example, if a label (text label) is created, it is placed in the desired frame by specifying a row and a column, as can be seen in Figure 6.1. A simplified flowchart of the GUI and the main code for the layout and options can be seen in Appendix 7.6. All of the code for the GUI is not appended, as it is about 1500 lines of code.

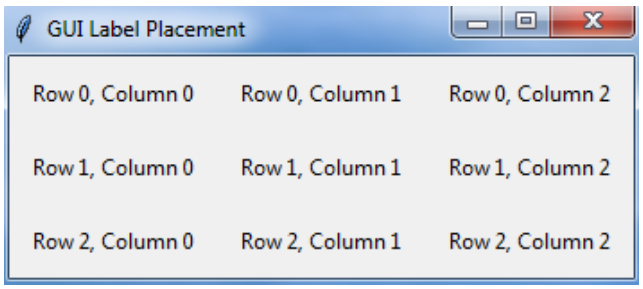


Figure 6.1: GUI Label Placement.

6.1.1 GUI Options

The GUI is developed in such a way that all of the functionality developed in the project, can be used through the GUI. All of the options the GUI provide, can be seen in Figure 6.2. The power limitation is set to 80 kW as default and the speed limitation is 120 km/h by default. Figure 6.3 shows an example of

the functionality of the GUI, by plotting the speed vs distance for the same route, power- and speed limitation as in Figure 5.4, but by making use of the GUI.

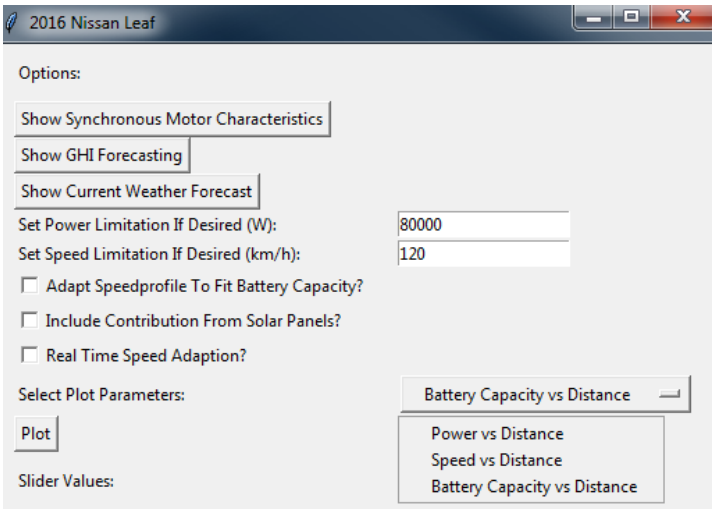


Figure 6.2: GUI Options.

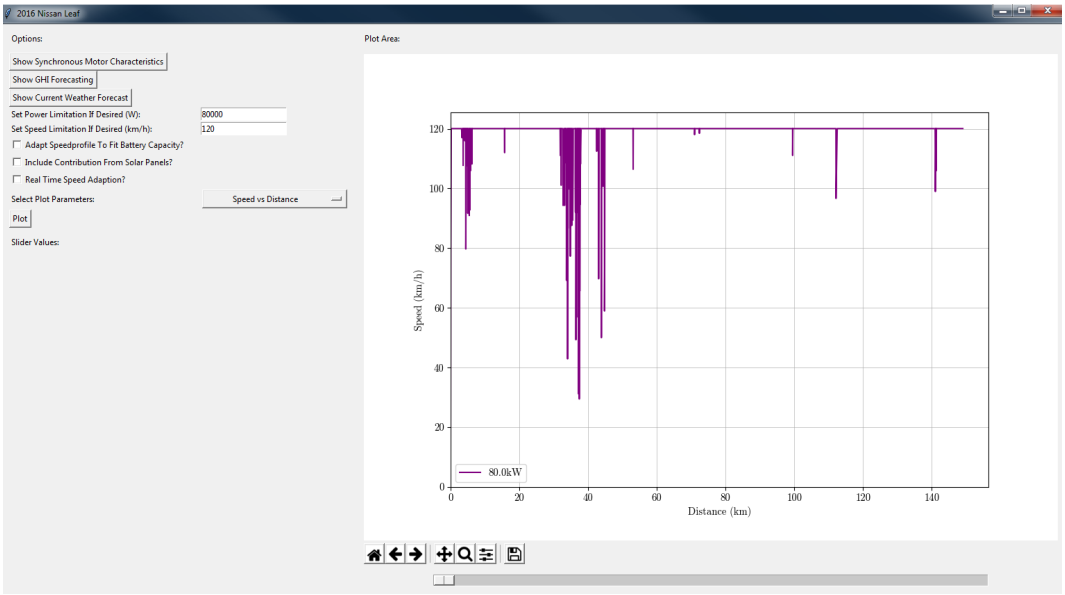
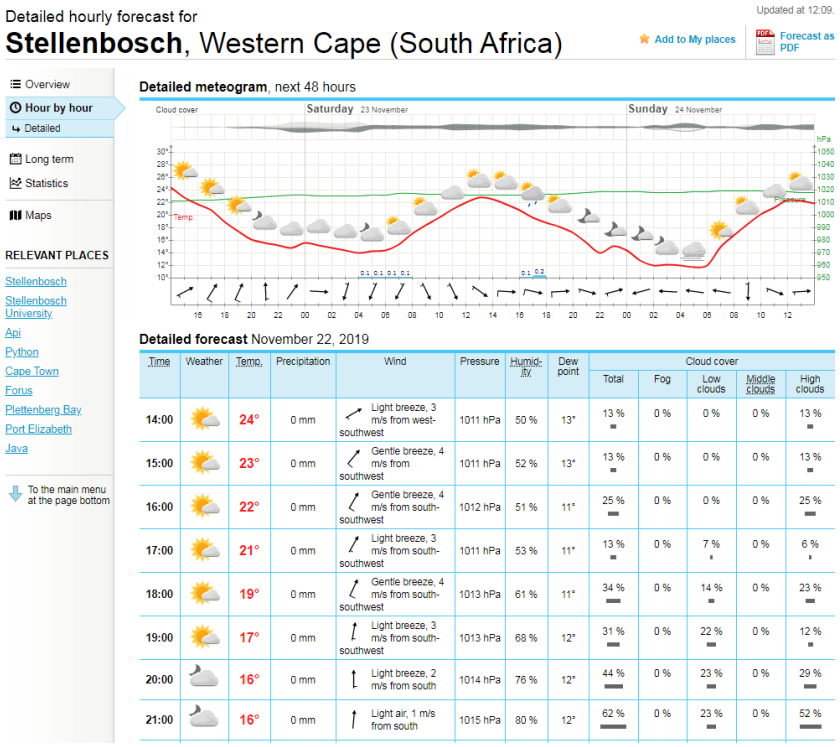


Figure 6.3: GUI Speed vs Distance.

When selecting the option of showing the current weather forecast, the GUI will take you to the web-page as seen in Figure 6.4.



Detailed hourly forecast for
Stellenbosch, Western Cape (South Africa)

Updated at 12:09.
Add to My places
Forecast as PDF

Overview
Hour by hour
Detailed
Long term
Statistics
Maps

RELEVANT PLACES
Stellenbosch
Stellenbosch University
Api
Python
Cape Town
Forus
Plettenberg Bay
Port Elizabeth
Java

To the main menu at the page bottom

Detailed meteogram, next 48 hours

Detailed forecast November 22, 2019

Time	Weather	Temp.	Precipitation	Wind	Pressure	Humidity	Dew point	Cloud cover				
								Total	Fog	Low clouds	Middle clouds	High clouds
14:00		24°	0 mm	Light breeze, 3 m/s from west-southwest	1011 hPa	50 %	13°	13 %	0 %	0 %	0 %	13 %
15:00		23°	0 mm	Gentle breeze, 4 m/s from southwest	1011 hPa	52 %	13°	13 %	0 %	0 %	0 %	13 %
16:00		22°	0 mm	Gentle breeze, 4 m/s from south-southwest	1012 hPa	51 %	11°	25 %	0 %	0 %	0 %	25 %
17:00		21°	0 mm	Light breeze, 3 m/s from south-southwest	1011 hPa	53 %	11°	13 %	0 %	7 %	0 %	6 %
18:00		19°	0 mm	Gentle breeze, 4 m/s from south-southwest	1013 hPa	61 %	11°	34 %	0 %	14 %	0 %	23 %
19:00		17°	0 mm	Light breeze, 3 m/s from south-southwest	1013 hPa	68 %	12°	31 %	0 %	22 %	0 %	12 %
20:00		16°	0 mm	Light breeze, 2 m/s from south	1014 hPa	76 %	12°	44 %	0 %	23 %	0 %	29 %
21:00		16°	0 mm	Light air, 1 m/s from south	1015 hPa	80 %	12°	62 %	0 %	23 %	0 %	52 %

Figure 6.4: Weather Forecast.

6.1.1.1 Adapt Speed Profile To Fit Battery Capacity

If the option is selected for the GUI to adapt the speed profile to fit the battery capacity, the following will happen: For a specific route, if the user sets a power limitation (which corresponds with a certain speed profile) that results in the battery being drained before the destination is reached, the power limitation is automatically decreased incrementally until a speed profile is found that will result in the route being able to be completed. An example of this functionality can be seen in Figure 6.5. The simulation is done for the same route as in Figure 5.6, but as seen in Figure 6.5, the EV is now able to complete the route. The power limitation that corresponds with this speed profile, is found to be 50 kW.

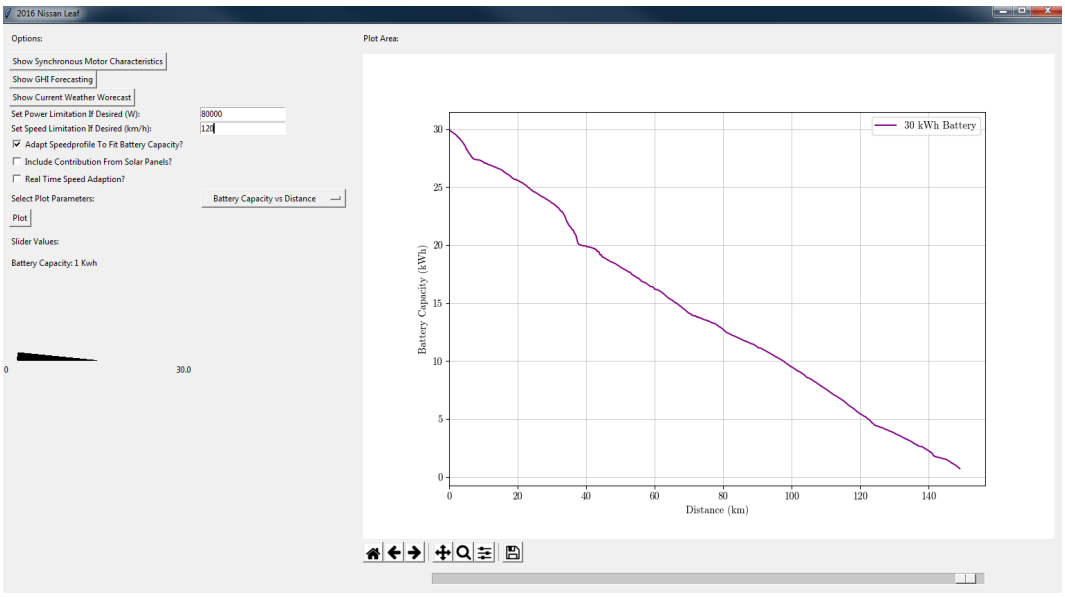


Figure 6.5: Adapt Speed Profile To Fit Battery Capacity.

6.1.2 Interactive Plots

When selecting the option to plot one of the available results, a slider bar is created right below the plot. This slider bar corresponds with the x-axis of the selected plot. The slider bar can be moved to select an x-value and the corresponding y-value(s) will be displayed by a gauge(s), as can be seen in Figure 6.6. The gauge(s) is updated in real time as the slider is moved. The figure that is plotted can be zoomed, as well as saved from within the GUI.

The coding for the gauge(s) is done in a simple but effective manner. With the maximum y-value(s) known, from the x-value selected by the slider bar, the ratio between the selected y-value and the maximum y-value is calculated. As the gauge is a circle that extends from 0° to 180°, the calculated ratio is multiplied by 180° to determine how far the circle has to extend for the current slider position. The gauge is then plotted by drawing a circle that extends to the calculated degrees, as seen in Figure 6.6. The corresponding y-value(s) is printed above the cosponsoring gauge(s), and also updates as the slider is moved. The flowchart describing the working of the slider bar can be seen in Figure 6.7.

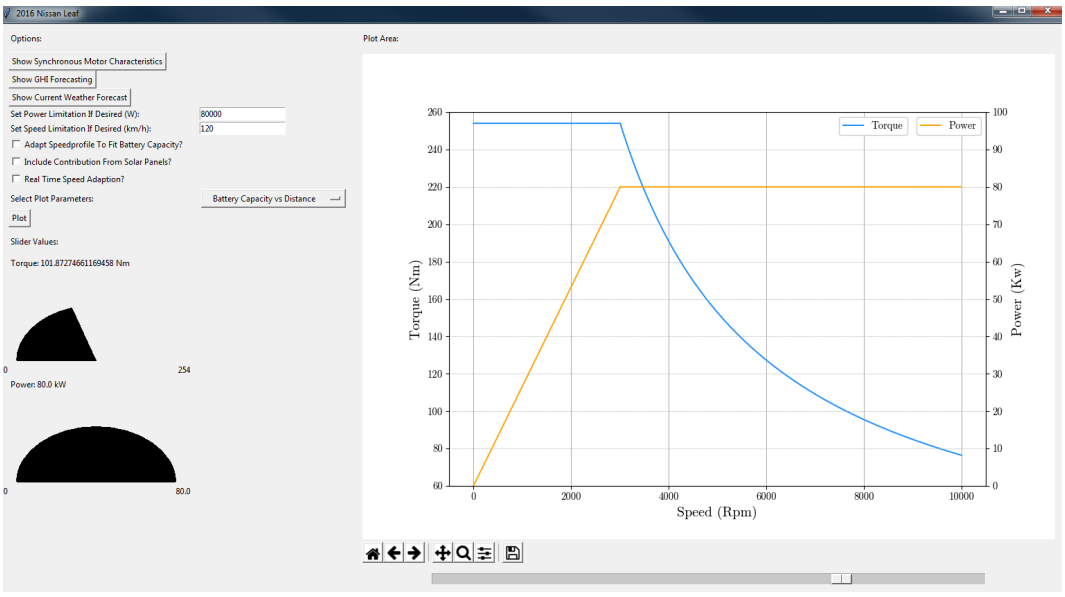


Figure 6.6: Motor Characteristics GUI.

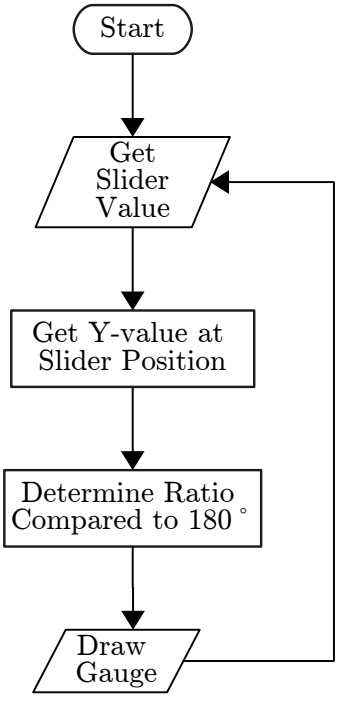


Figure 6.7: Slider Bar Flowchart.

6.1.3 Error Messages

For the speed limit and power limitation, the user has the option of entering a value. For the case where the user tries to 'break' the GUI by entering words where numbers are required, the try-except function is used to filter the input. If the input is invalid, a pop-up window will tell the user the entry is invalid and a new input can be entered. An example of the so called error message can be seen in Figure 6.8.

Another error message that can occur is when the user sets the desired power limitation, but this power limitation may be too small to overcome all of the opposing forces and will not be able to move forward. If this is the case, a pop-message similar to the one in Figure 6.8 will appear and inform the user that the desired power limitation is too small. The user will then be able to increase the power limitation.

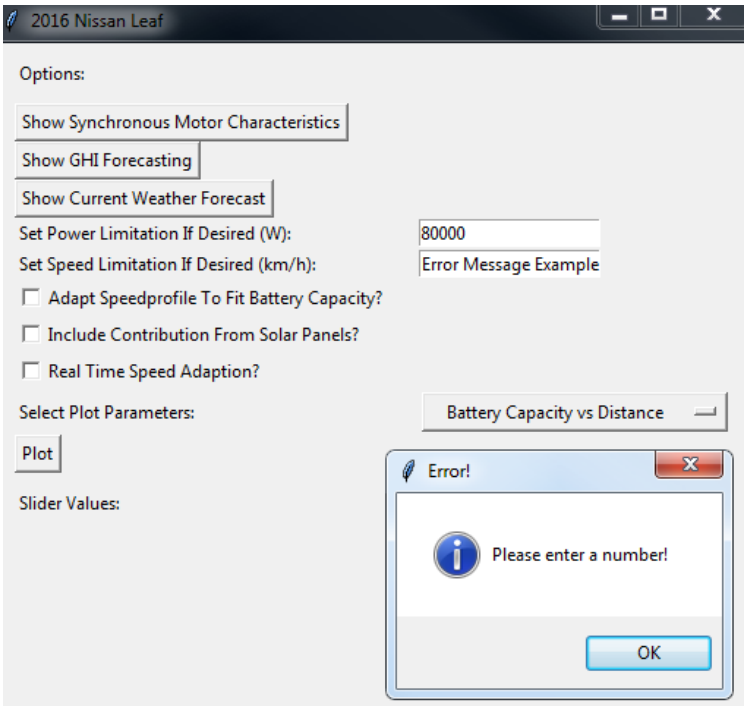


Figure 6.8: GUI Error Message.

6.2 PV Contribution While Driving

To review the contribution from the PV panels as the EV drives, three GHI intensities were compared, namely 1200 W/m^2, 800 W/m^2, 400 W/m^2 and 0 W/m^2. For the case where the GHI is 0 W/m^2, will be for the scenario where the EV is driving when the sun has set. It was modelled that the EV drives at four different constant-inclines. The ambient temperature for the simulation was set as 25 $°C$ and no power limitation was implemented, but the speed was limited to 80 km/h. For the case of no PV contributions, the weight of the panels were not brought into consideration, as it is for the scenario where there are no PV panels mounted on the PV. The result of these simulations can be seen in Table 6.1.

For the ideal scenario where the EV is driving on a road with a 0% incline and with optimal GHI, the range will be extended by 32 km, from PV contributions as the EV drives. When the PV panels receive no GHI, the weight of the two PV panels will cause the range of the EV to be slightly less than for when there are no PV panels mounted on the EV. It would be a very good idea if the PV panels can be mounted in such a way that they can be unplugged and removed with ease. An example would be if the PV panels are mounted on rails and then held in placed with end-clamps.

Table 6.1: EV Range

Road Incline	**0%**	**5%**	**10%**	**15%**
No PV Panels	309 km	100 km	60 km	44 km
PV @ 1200 W/m^2	341 km	100 km	60 km	43 km
PV @ 800 W/m^2	328 km	99 km	59 km	43 km
PV @ 400 W/m^2	316 km	98 km	58 km	43 km
PV @ 0 W/m^2	305 km	97 km	58 km	43 km

6.3 PV Contribution While Parked

For a scenario where the driver of the EV has a job with working hours of 08:00 to 17:00, the EV will be or can be parked in direct sunlight for 9 hours. In this time, the power generated by the PV panels can be used to charge the battery of the EV. The question poses, how much would the battery be able to charge within this time? To review this, a clear day and an overcast day were chosen for simulations, and the GHI of these two days can be seen in Figure 6.9. It is assumed that the vehicle is parked in direct sunlight, that there are no shading over the panels, there is no wind and the ambient temperature is

25 $^{\circ}C$. The location for the simulation is central Stellenbosch. For the clear day, the battery was charged 14% and for the overcast day the battery was charged 6%. Table 6.2 shows how far the EV will be able to drive with only the charge gained from being parked, for different road inclines.

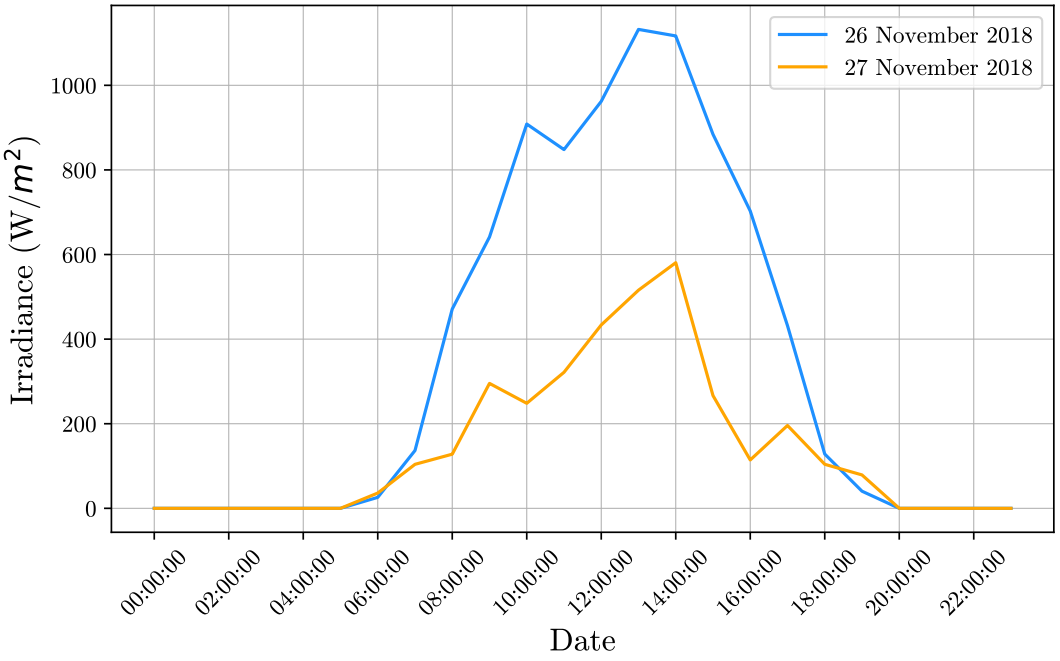


Figure 6.9: Simulation GHI Profile.

Table 6.2: EV Range from Battery Charged by PV only

Road Incline	**0%**	**5%**	**10%**	**15%**
Clear day	43 km	14 km	8 km	6 km
Overcast day	18 km	6 km	4 km	3 km

6.4 Route Planning

As a practical example of all the different components that make up this project, it was decided to plan a route from Stellenbosch to Bloemfontein. It was decided to do this, as it is a real-world example where this project can be implemented. As the trip from Stellenbosch to Bloemfontein is 1000 km, the EV will not be able to make the trip in one go, but will have to stop in order to recharge several times. Luckily, South Africa does currently have a few EV charging stations throughout the country. Figure 6.10 shows the EV charging stations along the route [74]. The route planning will be done for a clear- and an overcast day and will be the same days as seen in Figure 6.9.

In real-life applications, the GHI forecasting will be used to determine the predicted GHI values for the trip. The reason for not using the GHI forecasting in this example, is to investigate what effect a clear- and cloudy day will have on the PV yield, and not the accuracy of the GHI forecasting. The accuracy of the GHI forecasting is discussed in detail in Chapter 4. For comparison, the simulation will also include the instance where the EV has no PV panels mounted on it. The weight of the driver and the luggage was set as 100 kg.

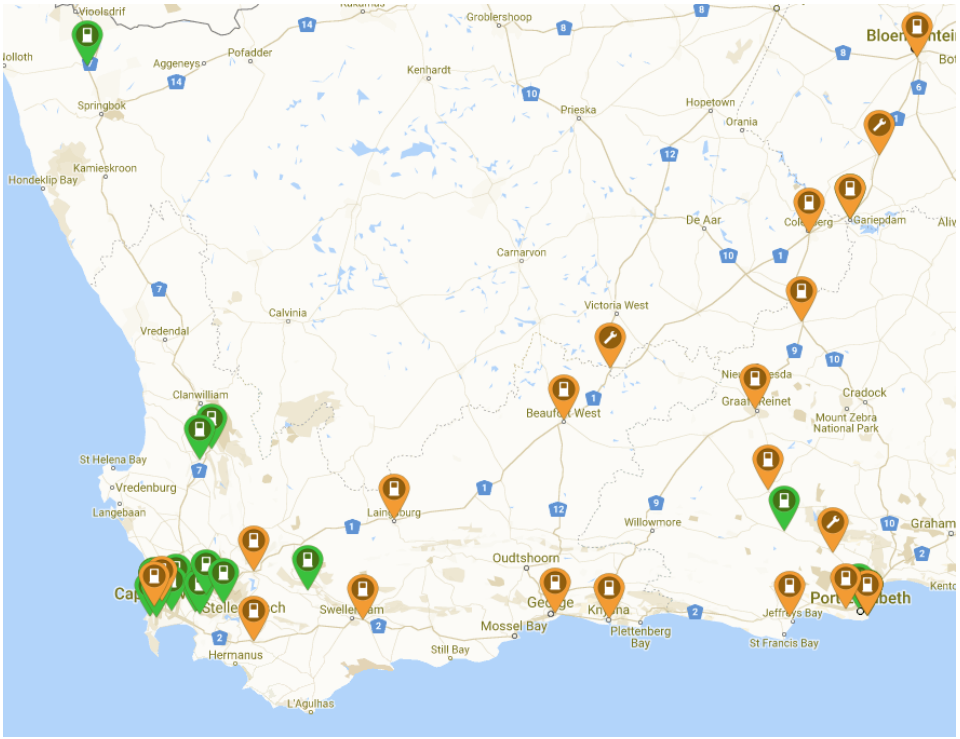


Figure 6.10: Charging Stations Along Route [74].

The 2016 Nissan Leaf supports fast-charge that can charge the battery from empty to 80% in roughly 30 minutes, at the charging stations that support

fast charge. When the EV is charged under normal-charge mode, the charge time will be about 6 hours.[75] The route is planned in such a way that the EV will stop and charge at the charge station until 80% of the capacity, which is equal to 24 kWh of energy. Where there are no EV charging stations, it is decided that the battery will be charged to maximum capacity through the normal charge mode. The driver will have to make an overnight stop, where the EV will be charged to maximum capacity as well. Table 6.3 shows the different stops along the route, as well as the estimated time of arrival (ETA) and the estimated time of departure (ETD).

Table 6.3: Route Planning

	Location	**ETA**	**ETD**	**Charge***
Departure	Stellenbosch	-	06:00	100%
Stop 1	Worcester	07:00	07:40	80%
Stop 2	Laingsburg	09:40	10:20	80%
Stop 3	Prince Albert Weg Service Station	11:20	17:20	100%
Stop 4	Beaufort West	18:30	19:10	80%
Stop 5	Richmond - Overnight	21:30	08:00	100%
Stop 6	Colesberg	09:30	10:10	80%
Stop 7	Trompsburg	11:30	17:30	100%
Arrival	Bloemfontein	18:40	-	-

* Charge Level at Departure

Figures 6.11 to 6.18 show the battery capacity as well as the suggested driving speed versus distance for each leg of the trip. The algorithm discussed in Section 6.1.1.1, is used to find a power limitation that will result in the EV being able to reach the destination. The route is planned in such a way in order to reach the biggest towns or towns with EV charging stations along the route. If the suggested driving speed is not followed exactly, the energy requirements will no be exactly as simulated. When departing from Stellenbosch, Prince Albert, Richmond and Trompsburg, as can be seen in Figures 6.11, 6.14, 6.16 and 6.18, the battery will be fully charged. For the rest of the trips, the battery capacity is 24 kWh as the battery will be quick charged to 80%. The cost of charging the battery at the charging stations, is R5.88/kWh [74].

From Figures 6.11 to 6.18 it can be seen that the contribution from the PV panels have a very small contribution to the range of the EV (the battery capacity.) The contribution from the PV panels is at a peak when the EV is

driving close to midday, as the GHI is at a maximum. For this example of route planning, the route will be planned exactly the same, whether the EV is fitted with PV panels or not.

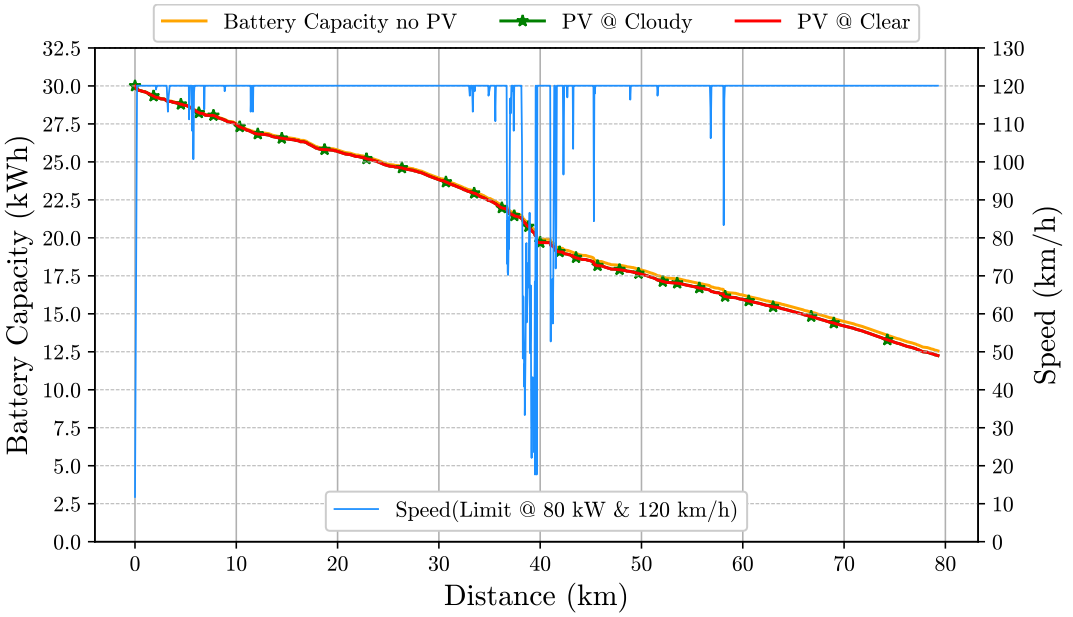


Figure 6.11: Stellenbosch to Worcester.

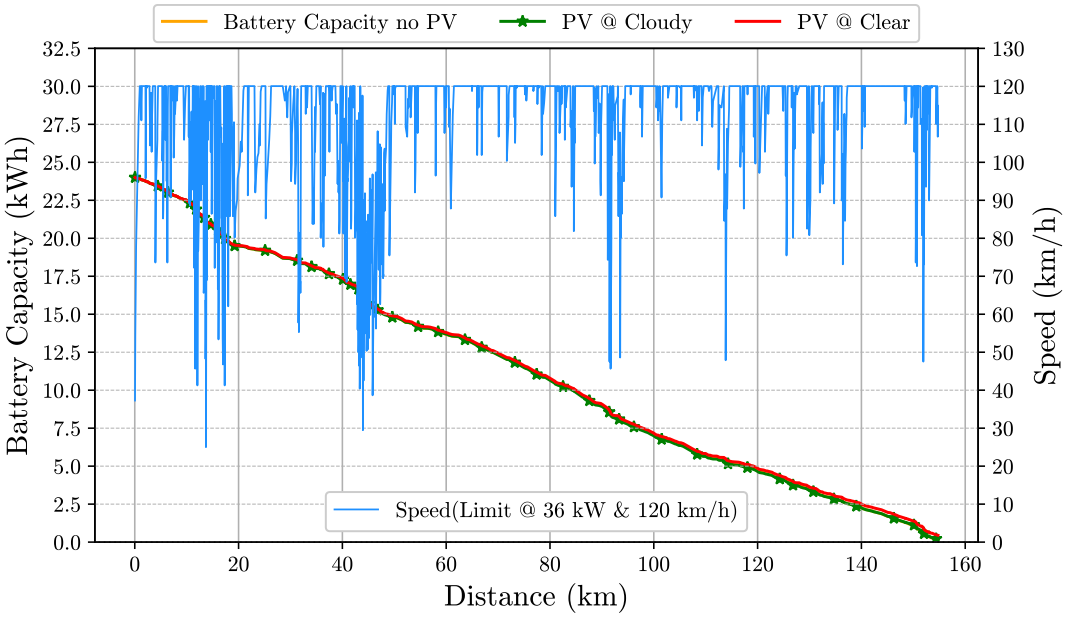


Figure 6.12: Worcester to Laingsburg.

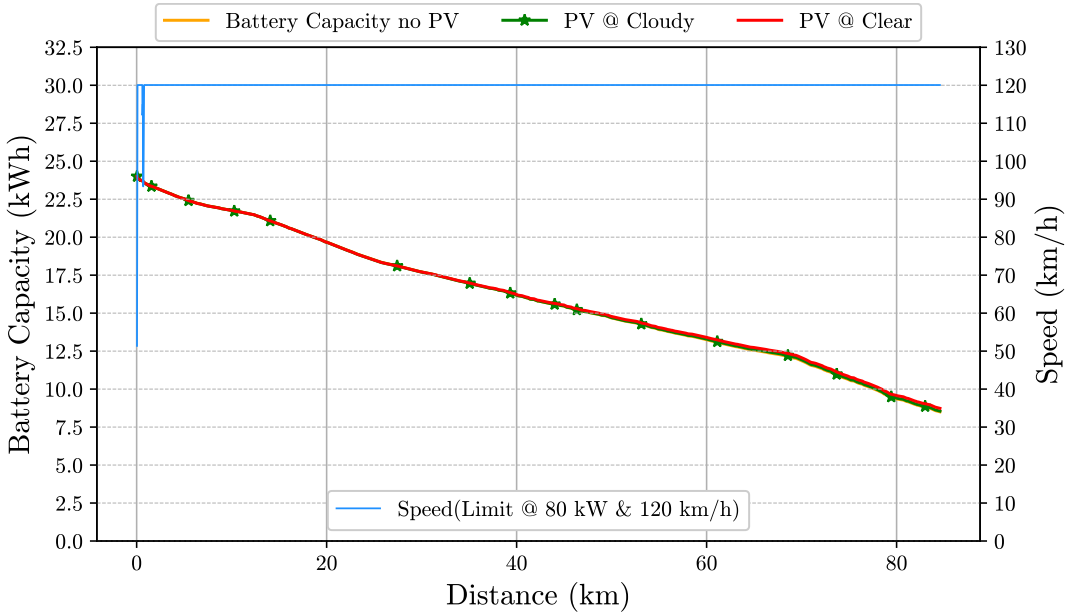


Figure 6.13: Laingsburg to Prince Albert.

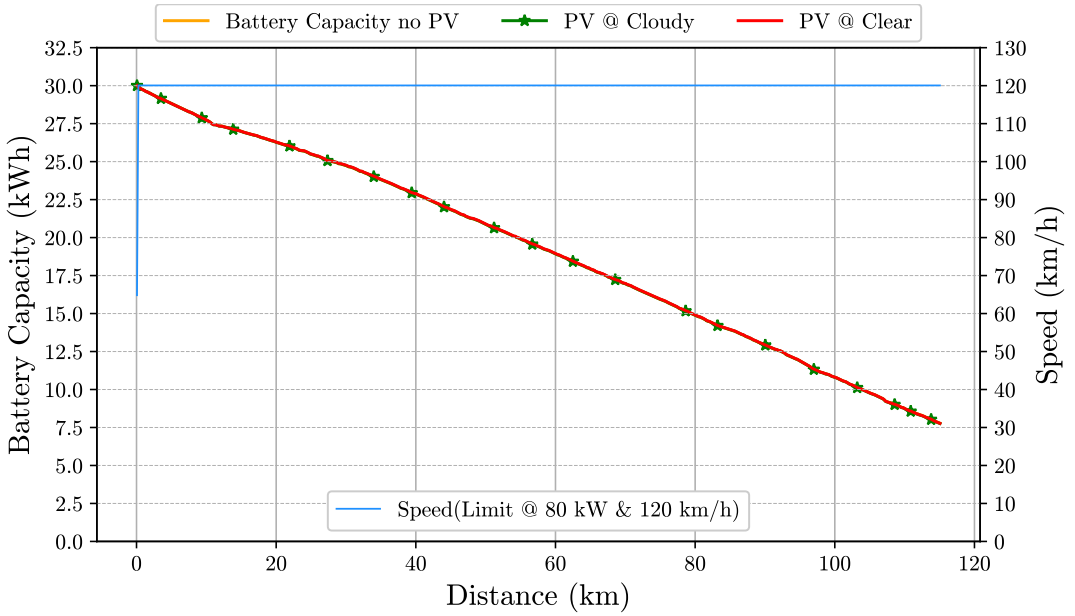


Figure 6.14: Prince Albert to Beaufort West.

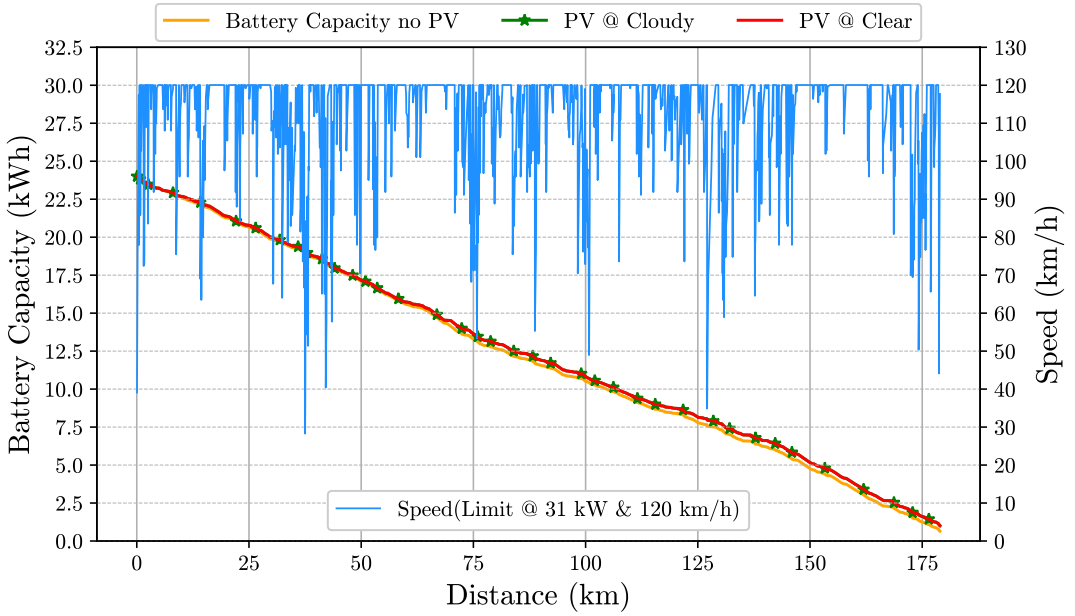


Figure 6.15: Beaufort West to Richmond.

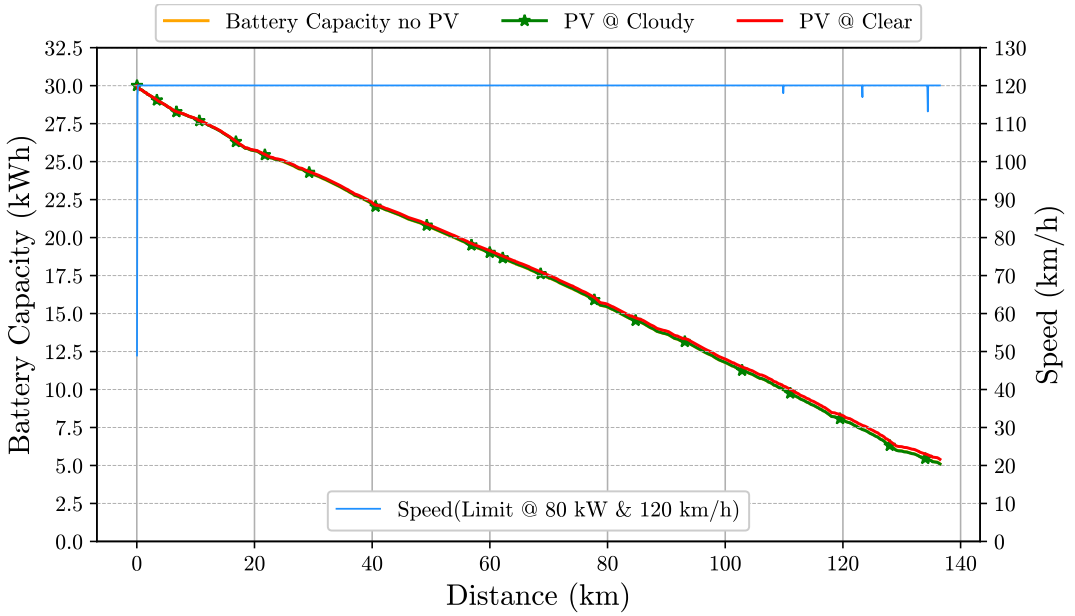


Figure 6.16: Richmond to Colesberg.

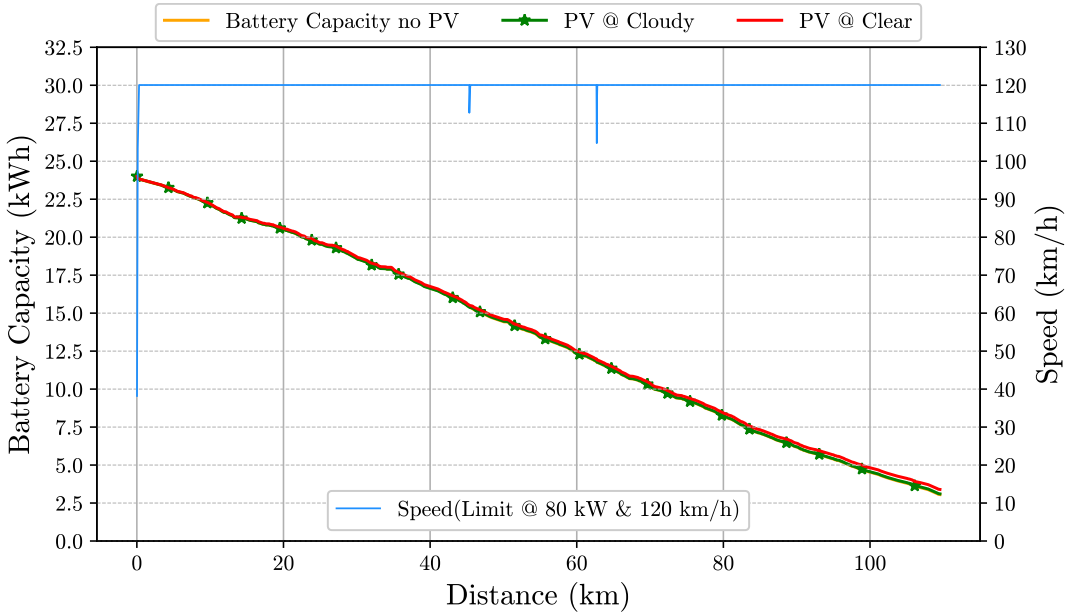


Figure 6.17: Colesberg to Trompsburg.

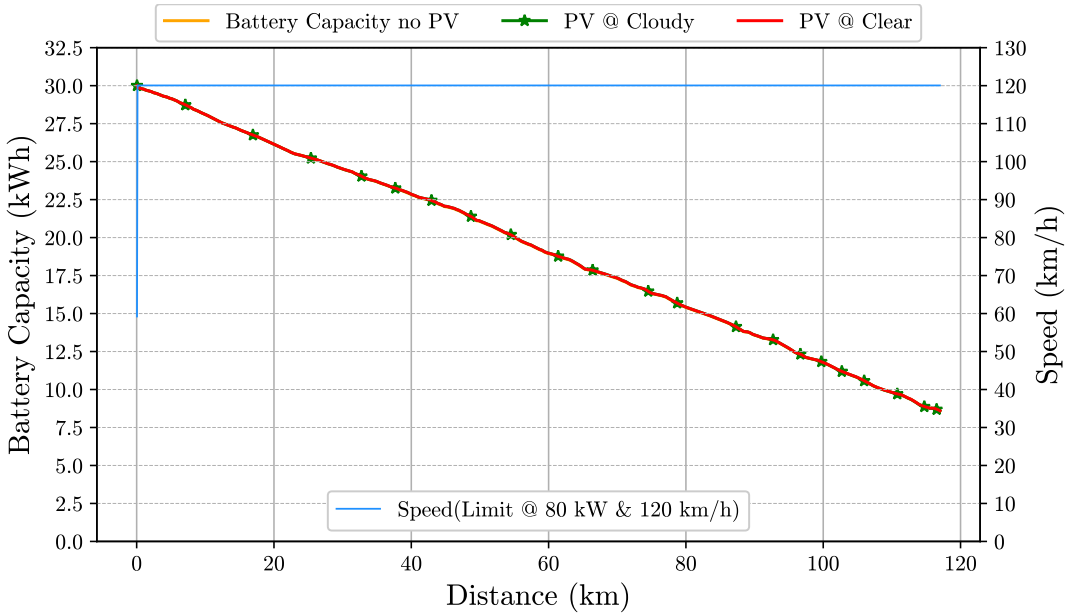


Figure 6.18: Trompsburg to Bloemfontein.

For comparative purposes, the route planning was done to determine how fast the journey can be completed, if the driver does not mind driving through the night and for the scenario where every stop will have an EV charging station. The departure and arrival times for the scenarios above, can be seen in Table 6.4. For the case where the driver drives through the night, the destination will

be reached 4 and a half hours earlier than with the overnight stop included. The driver can however sleep in the car for the duration where the EV has to be charged during night time. For the scenario where there are EV charging stations at every stop, the EV will be able to make the journey within one day.

Table 6.4: Route Planning Scenarios

	Location	**ETA**	**ETD**	**Charge***
No Overnight Stop				
Departure	Stellenbosch	-	06:00	100%
Stop 1	Worcester	07:00	07:40	80%
Stop 2	Laingsburg	09:40	10:20	80%
Stop 3	Prince Albert Weg Service Station	11:20	17:20	100%
Stop 4	Beaufort West	18:30	19:10	80%
Stop 5	Richmond	21:30	03:30	100%
Stop 6	Colesberg	05:00	05:40	80%
Stop 7	Trompsburg	07:00	13:00	100%
Arrival	Bloemfontein	14:10	-	-
EV Charging Station at Every Stop				
Departure	Stellenbosch	-	06:00	100%
Stop 1	Worcester	07:00	07:40	80%
Stop 2	Laingsburg	09:40	10:20	80%
Stop 3	Prince Albert Weg Service Station	11:20	12:00	80%
Stop 4	Beaufort West	13:10	13:50	80%
Stop 5	Richmond	16:10	16:50	80%
Stop 6	Colesberg	18:30	19:10	80%
Stop 7	Trompsburg	20:30	21:10	80%
Arrival	Bloemfontein	22:20	-	-

* Charge Level at Departure

Chapter 7

Conclusion and Recommendations

7.1 Electric Vehicles

This section covered the modelling of an EV, a 2016 Nissan Leaf to be specific. A dynamic mathematical model was created, that takes all of the forces that acts in on a vehicle into account. The section was concluded with a sensitivity analysis on the range of an EV, and it is found that the coefficient of drag and the road incline have the biggest effect on the range. Electric motors and batteries are also discussed briefly, as well as the modelling thereof. Both the electric motor and the battery were modelled very simplistically, but the option to expand the complexity of the modelling is discussed.

7.2 Solar Modelling

In this section, the whole process of determining the contribution from the PV panels is discussed. Firstly the different irradiance types are discussed, followed by the solar angles, in order to determine the total in-plane irradiance. The single diode model is discussed, along with the mathematical equations accompanying it. The losses regarding PV yield are discussed, and the losses are deemed negligible for this project. Shading and dust losses can be as high as 100% and 15% respectively, but is accounted for and discussed in this chapter. The section is concluded with a sensitivity analyses on the power output of a PV panel, with a change in windspeed and ambient temperature.

7.3 GHI Forecasting

This section covered GHI forecasting by making use of freely available data. The forecasting is based on the correlation between clearsky GHI, actual GHI and cloud cover. Different regression techniques were compared to see which is the most accurate. The results obtained were compared to an existing GHI

forecasting model by Larson et al [61]. It was found that polynomial regression was the most accurate, with a normalised RMSE of 44.22%, compared to 53.19% for the Larson model.

7.4 Software Development and Results

The physical algorithms that determine the range of the EV, the optimal driving speeds, battery capacity and contributions from the PV panels are discussed in this section. The detailed development of the road profile data is also discussed. It is concluded that it is not feasible to mount PV panels on an EV to extend the range of the EV while driving, as the range is extended only 10% under ideal conditions on a completely flat road. When the PV panels are fitted on the EV while driving at night, the extra weight of the panels will slightly decrease the range. For the case where the battery of the EV is charged by the PV panels while being parked, the outcome is a bit more positive. The EV can drive as far as 43 km using only the charge gained by being parked outside from 08:00 to 17:00, on a completely flat road. So if the driver stays relatively close to work, the trip home may cost the driver nothing.

7.5 Closing Remarks

It is concluded that the research objectives as defined in Section 1.4, are successfully met within the project. A 2016 Nissan Leaf is successfully modelled, and the factors that affect the range thereof were investigated. The PV panels that are mounted on the EV were modelled successfully, as well as the calculations in order to determine the total in-plane irradiance. The GHI forecasting showed promising results, and was done by making use of only freely available data. The algorithms to predict the EV range, give the driver optimal driving instructions and to determine the contributions from the PV panels, were successfully developed. Lastly, the GUI was developed to be easy to understand and as interactive as possible. Overall, the project was completed successfully.

7.6 Recommendations

From the results obtained in this project, the following recommendations are made for further work:

- Developing a PV panel that is smaller and lighter will increase the effect of PV contribution to the range of the EV. If the panels are small enough, there could possibly fit more than two PV panels on the EV.

- Research regarding the construction of more EV charging stations in SA will help the public overcome their range anxiety. The duration of long trips will also be decreased if there are more charging stations.
- Regarding the GHI forecasting, if more data can be used for the 'training' data, the forecasting model is expected to be even more accurate.
- The range prediction of the EV is dependant on the accuracy of the road profile data. If the data retrieved from Google Earth is inaccurate, the predicted range will also be inaccurate. Higher frequency of available data points along the respective routes, will also aid in the range prediction to be as accurate as possible.

List of References

[1] Bischof-niemz, T., Calitz, J., Tazvinga, H., Fourie, R., Mushwana, C., Nkosi, M. and Naidoo, M.: The impact of a significantly increased electric vehicle fleet on South Africa's 2016 Integrated Resource Plan. *CSIR Report*, 2017.

[2] Landman, C. and Rix, A.J.: Performance Prediction for an Electric Vehicle. *Proceedings - 2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa, SAUPEC/RobMech/PRASA 2019*, pp. 538–543, 2019.

[3] Neubauer, J. and Wood, E.: The impact of range anxiety and home, workplace, and public charging infrastructure on simulated battery electric vehicle lifetime utility. *Journal of Power Sources*, vol. 257, pp. 12–20, 2014. ISSN 03787753. Available at: `http://dx.doi.org/10.1016/j.jpowsour.2014.01.075`

[4] Larminie, J. and Lowry, J.: *Electric Vehicle Technology Explained*. 2003. ISBN 0470851635.

[5] Howroyd, S. and Thring, R.: An electric vehicle model and validation using a Nissan Leaf: A Python-based object-oriented programming approach. *Advances in Mechanical Engineering*, vol. 10, no. 7, pp. 1–7, 2018. ISSN 16878140.

[6] Rix, A.J.: Design , Comparison and Experimental Evaluation of Non-Overlap Winding Radial Flux Permanent Magnet Hub Drives for Electric Vehicles. *Dissertation presented for the degree of Doctor of Philosophy in the Faculty of Engineering at Stellenbosch University*, 2011.

[7] Asamer, J., Graser, A., Heilmann, B. and Ruthmair, M.: Sensitivity analysis for energy demand estimation of electric vehicles. *Transportation Research Part D: Transport and Environment*, vol. 46, pp. 182–199, 2016. ISSN 13619209. Available at: `http://dx.doi.org/10.1016/j.trd.2016.03.017`

[8] CleanPNG: Cartoon Car. Available at: `https://www.cleanpng.com`

[9] Hlinovský, T.: Optimal Control of Mathematical Model of the Electrovehicle. 2015. ISSN 0305-764X. `arXiv:1707.09838v1`.

[10] Mathcentre: Newton ' S Second Law of Motion. 2005. Available at: `http://www.mathcentre.ac.uk/`

[11] Umans, S.D.: *Fitzgerald & Kingsley's Electric Machinery*. McGraw-Hill Education, 2013. ISBN 9780073380469.
Available at: https://books.google.co.za/books?id=REI9XwAACAAJ

[12] Moyer, E.J. and Chicago, U.: Basics on electric motors. p. 11, 2010.

[13] Mustafa, B.: Field Weakening Control of Interior Permanent Magnet Synchronous Motor Employing Model Order Reduction. *Thesis submitted for examination for the degree of Master of Science in Technology.*, 2018.

[14] Knight, A.: Electric Machines. 2019.
Available at: https://people.ucalgary.ca/{~}aknigh/electrical{_}machines/index.html

[15] Albright, G., Edie, J. and Al-Hallaj, S.: A Comparison of Lead Acid to Lithium-ion in Stationary Storage Applications Published by AllCell Technologies LLC. , no. March, p. 14, 2012.
Available at: https://www.batterypoweronline.com/wp-content/uploads/2012/07/Lead-acid-white-paper.pdf

[16] Chen, H., Cong, T.N., Yang, W., Tan, C., Li, Y. and Ding, Y.: Progress in electrical energy storage system: A critical review. *Progress in Natural Science*, vol. 19, no. 3, pp. 291–312, 2009. ISSN 10020071.
Available at: http://dx.doi.org/10.1016/j.pnsc.2008.07.014

[17] Deng, D.: Li-ion batteries: Basics, progress, and challenges. *Energy Science and Engineering*, vol. 3, no. 5, pp. 385–418, 2015. ISSN 20500505.

[18] Yang, F., Wang, D., Zhao, Y., Tsui, K.L. and Bae, S.J.: A study of the relationship between coulombic efficiency and capacity degradation of commercial lithium-ion batteries. *Energy*, vol. 145, pp. 486–495, 2018. ISSN 03605442.
Available at: https://doi.org/10.1016/j.energy.2017.12.144

[19] Suyabodha, A.: A Relationship between Tyre Pressure and Rolling Resistance Force under Different Vehicle Speed. vol. 12004, pp. 10–14, 2017. ISSN 2261236X.

[20] Badran, O., Abdulhadi, E. and Mamlook, R.: Evaluation of parameters affecting wind turbine power generation. *Power*, , no. May 2014, pp. 1–8, 2003.

[21] König, W.: Die Tops und Flops im Windkanal. 2008.
Available at: https://www.autobild.de/artikel/autos-im-windkanal-666465.html

[22] Ludvigsen, K.: Turbine Speed with Style. 2006.
Available at: https://www.hemmings.com/blog/article/turbine-speed-with-style/

[23] Dow, J.: Tesla's Next-Gen Roadster. 2017.
Available at: https://electrek.co/2017/11/20/teslas-next-gen-roadster-technical-analysis/

[24] Masters, G.M.: *Renewable and effecient electric power systems*. 2nd edn. 2013.

[25] Abood, A.A.: A comprehensive solar angles simulation and calculation using matlab. *INTERNATIONAL JOURNAL OF ENERGY AND ENVIRONMENT*, vol. 6, no. 4, pp. 367–376, 2015.

[26] Ammonit Solar Monitoring Systems: Solar Resource Assesment. 2013.
Available at: `www.eko-eu.com`

[27] Blanc, P., Espinar, B., Geuder, N., Gueymard, C., Meyer, R., Pitz-Paal, R., Reinhardt, B., Renné, D., Sengupta, M., Wald, L. and Wilbert, S.: Direct normal irradiance related definitions and applications: The circumsolar issue. *Solar Energy*, vol. 110, no. October, pp. 561–577, 2014. ISSN 0038092X.

[28] Maxwell, E.L.: A quasi-physical model for converting hourly Global Horizontal to Direct Normal Insolation. *Solar Energy Research Institute*, , no. SERI/TR-215-3087, pp. 35–46, 1987.
Available at: `http://rredc.nrel.gov/solar/pubs/PDFs/TR-215-3087.pdf`

[29] Perez, R.R., Ineichen, P., Maxwell, E.L., Seals, R.D. and Zelenka, A.: Dynamic global-to-direct irradiance conversion models. *ASHRAE Transactions*, vol. 98, no. pt 1, pp. 354–369, 1992. ISSN 00012505.

[30] Reno, M.J., Hansen, C.W. and Stein, J.S.: Global Horizontal Irradiance Clear Sky Models: Implementation and Analysis. *SANDIA REPORT SAND2012-2389 Unlimited Release Printed March 2012*, , no. March, pp. 1–66, 2012.

[31] Hejase, H.A.N. and Assi, A.H.: Estimation of Global and Diffuse Horizontal Irradiance in Abu Dhabi, United Arab Emirates. In: *Renewable Energy in the Service of Mankind Vol II*, August, pp. 3–14. 2016.

[32] Vignola, F.: GHI correlations with DHI and DNI and the effects of cloudiness on one-minute data. *World Energy Forum, American Solar Energy Society, Colorado Renewable Energy Society*, , no. June 2012, 2012.
Available at: `http://ases.conference-services.net/resources/252/2859/pdf/SOLAR2012{_}0217{_}fullpaper.pdf`

[33] Ecomsart: PV System Performance: GHI to PoA. 2019.
Available at: `https://ecosmartsun.com/pv-system-performance-3/pv-system-performance-ghi-to-poa/`

[34] Loutzenhiser, P.G., Manz, H., Felsmann, C., Strachan, P.A., Frank, T. and Maxwell, G.M.: Empirical validation of models to compute solar irradiance on inclined surfaces for building energy simulation. *Solar Energy*, vol. 81, no. 2, pp. 254–267, 2007. ISSN 0038092X.

[35] King, D.L., Kratochvil, J.A. and Boyson, W.E.: Measuring solar spectral and angle-of-incidence effects on photovoltaic modules and solar irradiance sensors. In: *Conference Record of the IEEE Photovoltaic Specialists Conference*, pp. 1113–1116. 1997. ISBN 0780337670. ISSN 01608371.

[36] Mohammad, N. and Karim, T.: Design and implementation of hybrid automatic solar-tracking system. *Journal of Solar Energy Engineering, Transactions of the ASME*, vol. 135, no. 1, pp. 1–6, 2013. ISSN 01996231.

[37] Wald, L. and Wald, L.: Basics in Solar Radiation At Earth Surface. 2018. Available at: `https://hal-mines-paristech.archives-ouvertes.fr/hal-01676634/document`

[38] Nou, J., Chauvin, R., Thil, S. and Grieu, S.: A new approach to the real-time assessment of the clear-sky direct normal irradiance. *Applied Mathematical Modelling*, vol. 40, no. 15-16, pp. 7245–7264, 2016. ISSN 0307904X.

[39] Humboldt State University: Latitude and Longitude. 2015. Available at: `http://gsp.humboldt.edu/olm{_}2015/Courses/GSP{_}101/01{_}IntroAndReivew/Presentation/Latitude{_}and{_}Longitude.html`

[40] Honsberg, C. and Bowden, S.: Solar Time. 2019. Available at: `https://www.pveducation.org/pvcdrom/properties-of-sunlight/solar-time`

[41] Toothman, J. and Aldous, S.: How Solar Cells Work. *How stuff works*, 2000.

[42] Kumar, V., Patil, N. and Zope, B.: Solar Panel Selection for 5KW System A. Principle of a Photo-Voltaic Cell. , no. 4, pp. 62–65, 2017. Available at: `https://www.ijeat.org/wp-content/uploads/papers/v6i4/D4911046417.pdf`

[43] Maghami, M.R., Hizam, H., Gomes, C., Radzi, M.A., Rezadad, M.I. and Hajighorbani, S.: Power loss due to soiling on solar panel: A review. *Renewable and Sustainable Energy Reviews*, vol. 59, pp. 1307–1316, 2016. ISSN 18790690. Available at: `http://dx.doi.org/10.1016/j.rser.2016.01.044`

[44] Alrahim Shannan, N.M.A., Yahaya, N.Z. and Singh, B.: Single-diode model and two-diode model of PV modules: A comparison. *Proceedings - 2013 IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2013*, pp. 210–214, 2013.

[45] Tamrakar, V., S.C, G. and Sawle, Y.: Single-Diode and Two-Diode Pv Cell Modeling Using Matlab For Studying Characteristics Of Solar Cell Under Varying Conditions. *Electrical & Computer Engineering: An International Journal*, vol. 4, no. 2, pp. 67–77, 2015. ISSN 22015957. Available at: `http://www.wireilla.com/engg/ecij/papers/4215ecij07.pdf`

[46] Bonkoungou, D., Koalaga, Z. and Njomo, D.: Modelling and Simulation of photovoltaic module considering single-diode equivalent circuit model in MATLAB. *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 3, pp. 493–502, 2013. ISSN 2079-9292.

[47] King, D.L., Boyson, W.E. and Kratochvill, J.A.: SANDIA Report Photovoltaic Array Performance Model. , no. December, p. 43, 2004.
Available at: `http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0{#}online`

[48] Efficiency, H. and Module, P.: High efficiency poly module cs6x-340|345|350|355p-fg. , no. 1, 2001.

[49] Ekici, S. and Kopru, M.A.: Investigation of PV system cable losses. *International Journal of Renewable Energy Research*, vol. 7, no. 2, pp. 807–815, 2017. ISSN 13090127.

[50] Salem, F. and Awadallah, M.A.: Detection and assessment of partial shading in photovoltaic arrays. *Journal of Electrical Systems and Information Technology*, vol. 3, no. 1, pp. 23–32, 2016. ISSN 23147172.
Available at: `http://dx.doi.org/10.1016/j.jesit.2015.10.003`

[51] Maghami, M.R., Hizam, H., Gomes, C., Radzi, M.A., Rezadad, M.I. and Hajighorbani, S.: Power loss due to soiling on solar panel: A review. *Renewable and Sustainable Energy Reviews*, vol. 59, pp. 1307–1316, 2016. ISSN 18790690.
Available at: `http://dx.doi.org/10.1016/j.rser.2016.01.044`

[52] Malamaki, K.N.D. and Demoulias, C.S.: Minimization of electrical losses in two-axis tracking pv systems. *IEEE Transactions on Power Delivery*, vol. 28, no. 4, pp. 2445–2455, 2013. ISSN 08858977.

[53] Heaney, M.B.: Electrical conductivity and resistivity. *Electrical Measurement, Signal Processing, and Displays*, , no. January 2003, pp. 7–1–7–14, 2003.

[54] Appelgren, P.: *Gigawatt Pulsed Power Technologies and Applications*. May. 2011. ISBN 9789174159622.

[55] Kim, H., Chen, H., Zhu, J., Maksimovic, D. and Erickson, R.: Impact of 1.2kV SiC-MOSFET EV traction inverter on urban driving. *WiPDA 2016 - 4th IEEE Workshop on Wide Bandgap Power Devices and Applications*, pp. 78–83, 2016.

[56] Hayes, J.G. and Davis, K.: Simplified electric vehicle powertrain model for range and energy consumption based on EPA Coast-down Parameters and Test Validation by Argonne national lab data on the Nissan leaf. *2014 IEEE Transportation Electrification Conference and Expo: Components, Systems, and Power Electronics - From Technology to Business and Public Policy, ITEC 2014*, pp. 1–6, 2014.

[57] Ineichen, P. and Perez, R.: A new airmass independent formulation for the linke turbidity coefficient. *Solar Energy*, vol. 73, no. 3, pp. 151–157, 2002. ISSN 0038092X.

[58] Johnson, F.S.: The Solar Constant. *Journal of Meteorology*, vol. 11, no. 6, 1954.

[59] Kasten, F. and Young, A.T.: Revised optical air mass tables and approximation formula. *Applied Optics*, vol. 28, no. 22, p. 4735, 1989. ISSN 0003-6935.

[60] Ray, S.: 7 Regression Techniques you should know! 2015.
Available at: https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/

[61] Larson, D.P., Nonnenmacher, L. and Coimbra, C.F.: Day-ahead forecasting of solar power output from photovoltaic plants in the American Southwest. *Renewable Energy*, vol. 91, pp. 11–20, 2016. ISSN 18790682.
Available at: http://dx.doi.org/10.1016/j.renene.2016.01.039

[62] Graham, S.: Clouds & Radiation. 1999.
Available at: https://earthobservatory.nasa.gov/features/Clouds

[63] Menon, A.: Linear Regression Using Least Squares. 2018.
Available at: https://towardsdatascience.com/linear-regression-using-least-squares-a4c3456e8570

[64] Rawlings, J.O., Pantula, S.G. and Dickey, D.A.: *Applied Regression Analysis*, vol. 2. 1998. ISBN 0387984542.

[65] Umam, A.: Polynomial Regression Using Least Square Estimation. 2017.
Available at: https://ardianumam.wordpress.com/2017/09/21/polynomial-regression-using-least-square-estimation/

[66] Saraswat, M.: Practical Guide to Logistic Regression Analysis in R. 2019.
Available at: https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/logistic-regression-analysis-r/tutorial/

[67] learn developers, S.: sklearn.preprocessing.LabelEncoder. 2019.
Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

[68] Claeskens, G. and Hjort, N.L.: Model selection and model averaging. Tech. Rep., Cambridge University Press, 2008.

[69] Vandeput, N.: Forecast KPI: RMSE, MAE, MAPE & Bias. 2019.
Available at: https://medium.com/analytics-vidhya/forecast-kpi-rmse-mae-mape-bias-cdc5703d242d

[70] Ruiz, G.R. and Bandera, C.F.: Validation of calibrated energy models: Common errors. *Energies*, vol. 10, no. 10, 2017. ISSN 19961073.

[71] Veness, C.: Calculate distance, bearing and more between Latitude/Longitude points. 2019.
Available at: https://www.movable-type.co.uk/scripts/latlong.html

[72] Bullock, R.: Great Circle Distances and Bearings Between Two Locations. *Mdt*, pp. 3–5, 2007.
Available at: http://www.dtcenter.org/met/users/docs/write{_}ups/gc{_}simple.pdf

[73] Bizee Energy Lens: kW and kWh Explained. 2019.
Available at: `https://www.energylens.com/articles/kw-and-kwh`

[74] PlugShare: PlugShare.
Available at: `https://www.plugshare.com/`

[75] Nissan North America Inc: 2016 LEAF. *Specification Sheet*, 2015.

Appendices

Solar Code

```
#-------------------------------------------------------------------------------
def Power_Out(GPoa, Temp, Panels, Wind):
    temp_eff = pvsystem.sapm_celltemp(GPoa, Wind, Temp,
                                      model='roof_mount_cell_glassback')
    if GPoa == 0:
        return 0

    else:
        module = pvsystem.retrieve_sam(name='CECMod')
                                    ['Canadian_Solar_CS6X_350M_FG']

        system = pvsystem.PVSystem(module_parameters=module
                                   modules_per_string=1)#surface tilt

        I_Ll, Io, R_s, R_sh, nNsVth \
            = (system.calcparams_desoto(effective_irradiance=GPoa,
                                        temp_cell=temp_eff['temp_cell']))

        single_diode_out = pvsystem.singlediode(I_Ll, Io, R_s, R_sh,
                                                nNsVth, ivcurve_pnts=100)

        I = single_diode_out['i_mp']
        P = single_diode_out['p_mp']#Power at maximum power-point
        V = single_diode_out['v_mp']

    return P*Panels

def DNI(time, GHI, S_Zenith):
    times = pd.DatetimeIndex([time,time])
    ghi = pd.Series([GHI, GHI], index=times)
    zenith = pd.Series([S_Zenith, S_Zenith], index=times)
    pressure = 93193
    dirint_data = irradiance.dirint(ghi, zenith, times, pressure=pressure)
    return dirint_data[0]


def DHI(time, Lat, Long, Alt, GHI, DNI):
    stb = Location(Lat, Long, tz='Africa/Johannesburg', altitude=Alt)
    times = pd.date_range(start=time, end=time)
    times_localized = times.tz_localize(stb.tz)
    Sun_Data = solarposition.get_solarposition(times_localized, stb.latitude,
                                               stb.longitude)

    angle = math.radians(Sun_Data['zenith'])
    Dhi = GHI - DNI*math.cos(angle)
    return Dhi
```

```
def G_POA(tilt, azim, time, Lat, Long, Alt, GHI, albedo):
    # tilt = math.degrees(tilt)
    stb = Location(Lat, Long, tz='Africa/Johannesburg', altitude=Alt)
    times = pd.date_range(start=time, end=time)
    times_localized = times.tz_localize(stb.tz)
    Sun_Data = solarposition.get_solarposition(times_localized, stb.latitude,
                                                stb.longitude)

    if float(Sun_Data['zenith']) < 90:
        aoi = irradiance.aoi(tilt, azim, Sun_Data['zenith'],
                             Sun_Data['azimuth'])
        ground_diff = irradiance.get_ground_diffuse(tilt, GHI, albedo)
        dni = DNI(time, GHI, float(Sun_Data['zenith']))
        dhi = DHI(time, Lat, Long, Alt, GHI, dni)
        dhi12 = irradiance.isotropic(tilt, dhi)
        GPOA = irradiance.poa_components(aoi, dni, dhi12, ground_diff)

        return float(GPOA['poa_global'])
    else:
        return 0
```

GHI Forecasting Code

```
#-------------------------------------------------------------------------------
def Poly_Reg(Train, Test):
    Train['Timestamp'] = pd.to_datetime(Train['Timestamp'])
    Test['Timestamp'] = pd.to_datetime(Test['Timestamp'])

    X_Train1 = Train[['LClouds','MClouds','HClouds','Clearsky_GHI']]
    X_Test1 = Test[['LClouds','MClouds','HClouds','Clearsky_GHI']]
    Y_Train = Train['Measured_GHI']
    Y_Plot = Test['Measured_GHI']
    Y_Plot = pd.DataFrame(Y_Plot)
    Y_Plot.index = pd.RangeIndex(len(Y_Plot.index))
    Y_Plot.index = range(len(Y_Plot.index))

    poly = PolynomialFeatures(degree = 2)
    X_poly = poly.fit_transform(X_Train1)
    X_poly_T = poly.fit_transform(X_Test1)
    poly.fit(X_poly, Y_Train)
    lin2 = LinearRegression()
    lin2.fit(X_poly, Y_Train)
    prediction1 = lin2.predict(X_poly_T)
    prediction1[np.where(X_Test1['Clearsky_GHI'] <= 0)] = 0
    prediction1[np.where(prediction1 < 0)] = 0
    y_pred = prediction1

    for i in range(len(y_pred)):
        if X_Test1['LClouds'].iloc[i] < 5 and X_Test1['MClouds'].iloc[i] < 5
            and X_Test1['HClouds'].iloc[i] < 5:
            y_pred[i] = X_Test1['Clearsky_GHI'].iloc[i]
        if X_Test1['Clearsky_GHI'].iloc[i] == 0:
            y_pred[i] = X_Test1['Clearsky_GHI'].iloc[i]
    y_pred = pd.DataFrame(y_pred, columns=['pred'])
    return y_pred, Y_Plot
```

```
def Logistic_Reg(Train, Test):
    Train['Timestamp'] = pd.to_datetime(Train['Timestamp'])
    Test['Timestamp'] = pd.to_datetime(Test['Timestamp'])

    X_Train1 = Train[['LClouds','MClouds','HClouds','Clearsky_GHI']]
    X_Test1 = Test[['LClouds','MClouds','HClouds','Clearsky_GHI']]
    Y_Train = Train['Measured_GHI']
    Y_Plot = Test['Measured_GHI']
    Y_Plot = pd.DataFrame(Y_Plot)
    Y_Plot.index = pd.RangeIndex(len(Y_Plot.index))
    Y_Plot.index = range(len(Y_Plot.index))

    LogReg = LogisticRegression(max_iter = 100, solver='lbfgs', random_state=0)
    lab_enc = preprocessing.LabelEncoder()
    training_scores_encoded = lab_enc.fit_transform(Y_Train)
    LogReg.fit(X_Train1, training_scores_encoded)
    y_pred = LogReg.predict(X_Test1)*4.5 #test to improve accuracy

    for i in range(len(y_pred)):
        if X_Test1['LClouds'].iloc[i] < 5 and X_Test1['MClouds'].iloc[i] < 5
            and X_Test1['HClouds'].iloc[i] < 5:
            y_pred[i] = X_Test1['Clearsky_GHI'].iloc[i]
        if X_Test1['Clearsky_GHI'].iloc[i] == 0:
            y_pred[i] = X_Test1['Clearsky_GHI'].iloc[i]
    y_pred = pd.DataFrame(y_pred, columns=['pred'])
    return y_pred, Y_Plot

def Linear_Regression(Train, Test):
    Train['Timestamp'] = pd.to_datetime(Train['Timestamp'])
    Test['Timestamp'] = pd.to_datetime(Test['Timestamp'])

    X_Train1 = Train[['LClouds','MClouds','HClouds','Clearsky_GHI']]
    X_Test1 = Test[['LClouds','MClouds','HClouds','Clearsky_GHI']]
    Y_Train = Train['Measured_GHI']
    Y_Plot = Test['Measured_GHI']
    Y_Plot = Test['Measured_GHI']
    Y_Plot = pd.DataFrame(Y_Plot)
    Y_Plot.index = pd.RangeIndex(len(Y_Plot.index))
    Y_Plot.index = range(len(Y_Plot.index))

    regr1 = linear_model.LinearRegression()
    regr1.fit(X_Train1, Y_Train)
    prediction1 = regr1.predict(X_Test1)
    prediction1[np.where(X_Test1['Clearsky_GHI'] <= 0)] = 0
    prediction1[np.where(prediction1 < 0)] = 0

    for i in range(len(prediction1)):
        if X_Test1['LClouds'].iloc[i] < 5 and X_Test1['MClouds'].iloc[i] < 5
            and X_Test1['HClouds'].iloc[i] < 5:
            prediction1[i] = X_Test1['Clearsky_GHI'].iloc[i]
        if X_Test1['Clearsky_GHI'].iloc[i] == 0:
            prediction1[i] = X_Test1['Clearsky_GHI'].iloc[i]

    y_pred = pd.DataFrame(y_pred, columns=['pred'])
    return y_pred, Y_Plot
```

```
def Larson(Final_Data_Test):
    Final_Data_Test['Timestamp'] = pd.to_datetime(Final_Data_Test['Timestamp']
    GHI = Final_Data_Test['Clearsky_GHI']*(0.35 +0.65*(1-Final_Data_Test['Clouds'
    return GHI
```

Algorithm Code

Road Profile Data

```
#-------------------------------------------------------------------------
def Dist_Between(lat1, lat2, lon1, lon2):
    R = 6371e3 # metres
    l1 = math.radians(lat1)
    l2 = math.radians(lat2)
    Δl1 = math.radians(lat2-lat1)
    Δl2 = math.radians(lon2-lon1)

    a = (math.sin(Δl1/2) * math.sin(Δl1/2) +
            math.cos(l1) * math.cos(l2) *
            math.sin(Δl2/2) * math.sin(Δl2/2))
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))

    d = R * c
    return d

def Heading(lat1, lat2, lon1, lon2):
    y = math.sin(lon2-lon1) * math.cos(lat2)
    x = (math.cos(lat1)*math.sin(lat2) -
            math.sin(lat1)*math.cos(lat2)*math.cos(lon2-lon1))
    brng = math.fabs(math.degrees(math.atan2(y, x)))
    return brng

def Elevation(lat1, lon1):
    response = requests.get(f'https://maps.googleapis.com/maps/api/elevation/
                            json?locations={lat1},{lon1}&key=AIzaSyB3vXddQwUm
                             -TwNk_v3CmW9OKn_WcqWjBc')
    data = response.json()
    print("Succes")
    norm = json_normalize(data['results'], sep="_")
    norm = pd.DataFrame(norm)
    return float(norm['elevation'])

Angle = np.tanh(Elevation/Dist_Between)
```

Power Requirements

```
#-------------------------------------------------------------------------
def Power_Needed_Lookup(speed, angle, acc, m):
        # Returns Power needed
        # Constants:
        urr = 0.0075            # Friction Coeffecient
        g = 9.81                # Gravity accelration
        p = 1.255               # Air density
        A = 2.744                 # Front area of vehicle
        Cd = 0.28                # Drag coeffecient
        G = 7.9377                  # Gear ratio
        n = 0.8                 # Gear effeciency
        r = 0.216                # Wheel radius

        constants = [urr, m, g, p, A, Cd, G, n, r]
        #            0    1  2  3  4  5   6  7  8
        temp_time = []
        for t in range(len(speed)):
            if t < 1:
                temp_time.append(0)
            else:
                # Determines time if speed stays constant
                if speed[t] == speed[t-1]:
                    temp_time.append (0.5/speed[t])
                else:
                    # Determines time when car is accelerating or decelerating
                    temp_time.append (np.abs((speed[t] - speed[t-1]))/acc)

        time = []
        # Creates time array 'incrementally'
        for k in range (len(angle)):
            if k < 1:
                time.append(0)
            else:
                time.append(temp_time[k]+ time[k-1])
        time = np.array(time)

        acceleration = []
        for l in range (len(angle)):
            if l < 1:
                # No acceleration at start
                acceleration.append(0)
            else:
                # Acceleration is delta speed over delta time
                acceleration.append((speed[l]-speed[l-1])/((time[l])-time[l-1]))
        acceleration = np.array(acceleration)

        temp_sin = []
        #  Sin of individual road segments, than append in array for Fhc
        for m in range (len(angle)):
            temp_sin.append(math.sin(angle[m]))
        temp_sin = np.array(temp_sin)
```

```
        Frr = constants[0]*constants[1]*constants[2#Rolling resistance
        Fad = 0.5*constants[3]*constants[4]*constants[5]*(speed**2) #Aerod drag
        Fhc = constants[1]*constants[2]*temp_sin           #Hill climbing force
        Fla = constants[1]*acceleration          #Newton's second law
        Fte = Frr + Fad + Fhc + Fla          #Total tractive effort

        T = (Fte*constants[8])/constants[6]             #Torque
        omega = speed/constants[8]              #Angular velocity
        P = T*omega*constants[6]                    #Power needed
        return P
```

```
#-------------------------------------------------------------------------------
def Power_Needed_Lookup_Prev_Speed(speed, prevspeed, angle, dist, acc, m):
        # Same as the Power_Needed function, except that it incorporates the
        # previous driving speed.
        #Constants:
        urr = 0.0075            #Friction Coeffecient
        g = 9.81                #Gravity accelration
        p = 1.255               #Air density
        A = 2.744                 #Front area of vehicle
        Cd = 0.28                #Drag coeffecient
        G = 7.9377                  #Gear ratio
        n = 0.8                 #Gear effeciency
        r = 0.216                #Wheel radius

        constants = [urr, m, g, p, A, Cd, G, n, r]
        #            0    1  2  3  4  5   6  7  8
        temp_time = []
        for t in range (len(speed)):
            if t < 1:
                temp_time.append (0)
            else:
                if speed[t] == prevspeed:
                    temp_time.append (dist/speed[t])

                else:
                    temp_time.append (np.abs((speed[t] - prevspeed))/acc)

        time = []
        for k in range (len(angle)):
            if k < 1:
                time.append(temp_time[k])
            else:
                time.append(temp_time[k]+ time[k-1])
        time = np.array(time)

        acceleration = []
        for l in range (len(angle)):
            if l < 1:
                acceleration.append(0)
            else:
                if prevspeed >= speed[l]:
                    acceleration.append(0)
                else:
                    const = [0.5*acc, 0, -dist]
                    r = np.roots(const)
                    acceleration.append((speed[l]-prevspeed)/r[0])
        acceleration = np.array(acceleration)

        temp_sin = []
        for m in range (len(angle)):
            temp_sin.append(math.sin(angle[m]))
        temp_sin = np.array(temp_sin)
```

```
Frr = constants[0]*constants[1]*constants[2] #Rolling resistance
        Fad = 0.5*constants[3]*constants[4]*constants[5]*(speed**2)#Aerodynamic
        Fhc = constants[1]*constants[2]*temp_sin            #Hill climbing force
        Fla = constants[1]*acceleration                  #Newton's second law
        Fte = Frr + Fad + Fhc + Fla          #Total tractive effort

        T = (Fte*constants[8])/constants[6]                  #Torque
        omega = speed/constants[8]                        #Angular velocity
        P = T*omega*constants[6]                         #Power needed
        return P
```

```
#-------------------------------------------------------------------------------
def find_nearest(array, value):
        #This function is used to find the nearest value in an array to
        # the desired value and is used for the acceleration cap in this case
        array = np.asarray(array)
        idx = (np.abs(array - value)).argmin()
        return array[idx]
```

Speed Instructions

```
#-------------------------------------------------------------------------------
def Speed_Original(power, angle, dist, acc, speed_limit, m)
        # This function is used to determine the maximum allowable speed for a
        # certain trip with a certain power-limitation. Inputs are the
        # power-limit, road prifile, distannce between data points of road
        # profile, maximum acceleration of the modeled vehicle, the
        # desired speed limit and vehicle mass.

        #array created to populate previous speed with
        previous = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
        previous = np.array(previous)

        # Creates the virtual lookup table according to the acceleration and
        # speed limit
        timet = (speed_limit/3.6)/acc
        dist_input = math.ceil(0.5*acc*(timet**2)) * 2
        speedbook = []
        for i in range(dist_input):
            const = [0.5*acc, 0, -i/2]
            r = np.roots(const)
            speedbook.append(r[0]*acc)

        speedbook = np.array(speedbook)

        V1 = []
        Power_1 = []
        Ptest1 = []

        for i in range(len(angle)):
            angle1 = []
            angle2 = []

            # Populates array with the angle of the road at the current point.
            # This is used to find the next speed as speedbook is the speed
            # profile, this angle is then the 'road profile'
            for z in range(len(speedbook)):
                angle1.append(angle[i])

            if i == 0:
                Vinp = 0
                power_available = power
                tempval = 0

            else:
                if i == 1:
                    Vinp = V1[i-1]

                # This just determines wheter the suggested speed will be
                # reached, and if notm what the speed will be reached
                if V1[i-2] < V1[i-1]:
                    c = [0.5*acc, V1[i-2], -dist[i]]
                    j = np.roots(c)
                    v  = j[j>0]
                    Vinp = (acc*v) + V1[i-2]
                    if Vinp > V1[i-1]:
                        Vinp = V1[i-1]
```

```
# Tests if the power is too small to complete the route, this
                # happens when two speeds are zero in a row
                if (V1[i-2] and V1[i-1]) == 0:
                    print("Power limitation to small to complete route!")
                    tkinter.messagebox.showinfo("Error!",
                            "Power limitation to small to complete route!")
                    return None, None, None, None

                else:
                    Vinp = V1[i-1]

                # Crates an array with the previous road angle, in order to
                # determine the power needed to maintain the speed of the
                # vehicle at the spesific speed
                for z in range(len(previous)):
                    angle2.append(angle[i-1])

                for t in range(len(previous)):
                    previous[t] = Vinp

                if Vinp == 0:
                    power_available = power
                    tempval = 0

                # Determines the power needed to maintain the speed of the
                # vehicle at the spesific speed and then determines the
                # power available for the next road angle
                else:
                    tempp = Power_Needed_Lookup(previous, angle2, acc, m)
                    tempval = tempp[2]
                    if tempval < 0:
                        power_available = power
                        tempval = 0
                    else:
                        power_available = power - tempval

            if power_available < 0:
                V1.append(0)
                Power_1.append(0)

            # This is the actual step where the virtual lookup table is created
            # with the power available at that point.
            else:
                PPP = Power_Needed_Lookup_Prev_Speed(speedbook,
                                                     Vinp, angle1, dist[i], acc, m)
                Ptesttemp = PPP - tempval
                Ptest2 = Ptesttemp[Ptesttemp < power_available]
                index = np.where(Ptesttemp == Ptest2[(len(Ptest2)-1)])
                const = [0.5*acc, 0, -dist[i]]
                r = np.roots(const)
                vvv = acc*r[0]
                acc_max = vvv
                if i == 0:
```

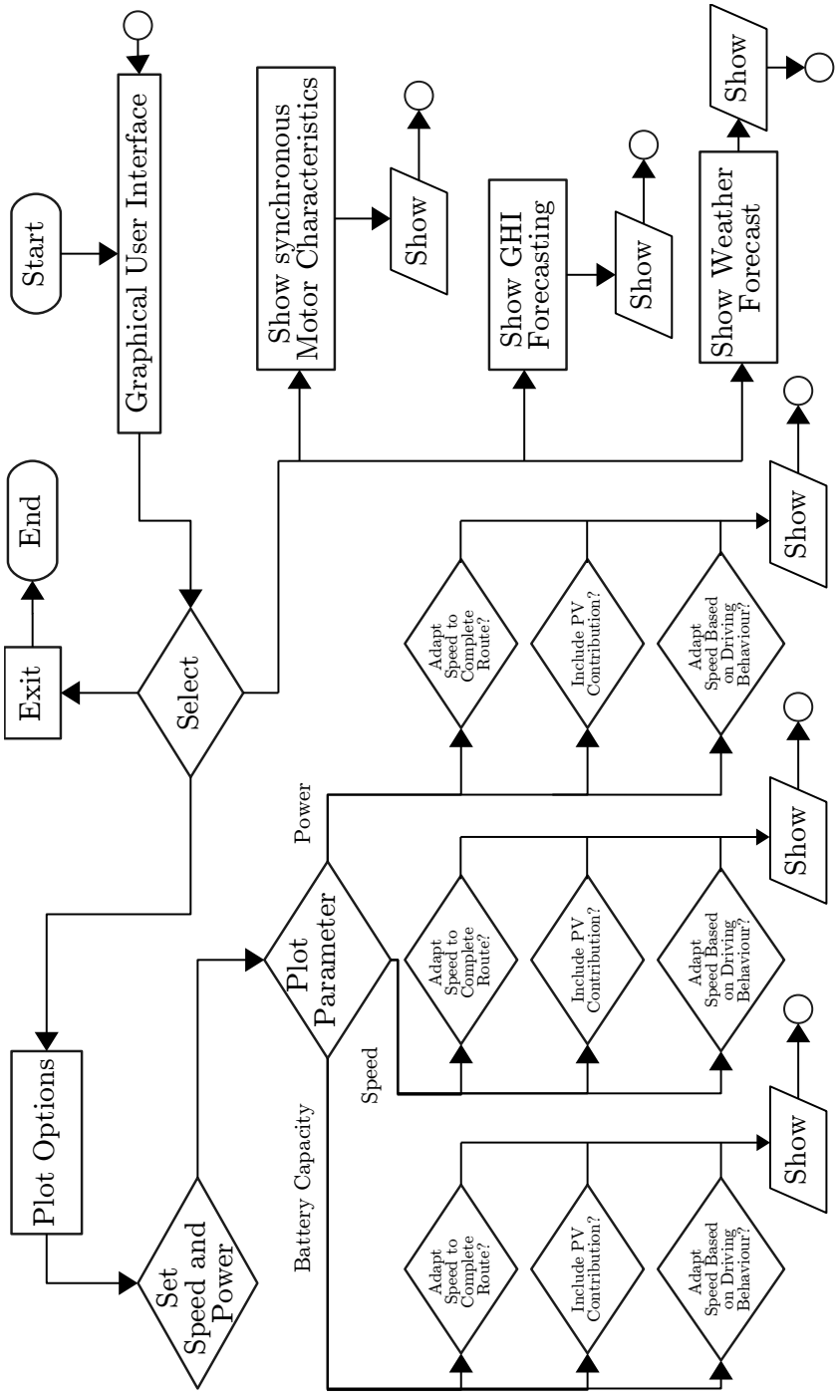
```
# Checks to see if the initail acceleration is bigger than
                    # allowed, as the model does not include factors such as
                    # tyre spin. If speed is to big, nearest function is
                    # used to find max allowable speed at the acceleration
                    # and the new 'index' is also determined
                    if speedbook[index] > acc_max:
                        V1.append(find_nearest(speedbook, acc_max))
                        index1 = np.where(speedbook == find_nearest(speedbook,
                                                                    acc_max))
                        Power_1.append(PPP[index1])
                    else:
                        Power_1.append(PPP[index])
                        V1.append(speedbook[index])

                # Checks to see if the initail acceleration is bigger than
                # allowed, as the model does not include factors such as tyre
                # spin. If speed is to big, nearest function is used to
                # find max allowable speed at the acceleration and the new
                # 'index' is also determined
                else:
                    boolean = speedbook[index] - V1[i-1]
                    if boolean > acc_max:
                        V1.append(V1[i-1] + find_nearest(speedbook, acc_max))
                        index1 = np.where(V1[i-1] + find_nearest(speedbook,
                                                                 acc_max))
                        Power_1.append(Ptesttemp[index1]+tempval)
                    else:
                        Power_1.append(Ptesttemp[index]+tempval)
                        V1.append(speedbook[index])

        V1 = np.array(V1)
        Power_1 = np.array(Power_1)
        Power_1[Power_1 < 0] = 0

        xval = []
        for t in range(len(angle)):
            if t == 0:
                xval.append(dist[t])
            else:
                xval.append(dist[t] + xval[t-1])
        xval = np.array(xval)

        return V1, Power_1, xval
```

Battery

```
#-------------------------------------------------------------------------
def Battery(speed, power, dist, capacity):

        # Determines the energy requiremtns for a route, knowing the speed
        # pattern and the power usage along the route.

        temp_time = []
        for t in range (len(speed)):
            if t < 1:
                temp_time.append((speed[t])/2.68)
            else:
                if speed[t] == speed[t-1]:
                    if speed[t] == 0:
                        temp_time.append(0)
                    else:
                        temp_time.append (dist[t]/speed[t])
                else:
                    if speed[t] > speed[t-1]:
                        temp_time.append(np.abs((speed[t] - speed[t-1]))/2.68)
                    else:
                        temp_time.append (dist[t]/speed[t])

        time = []
        for k in range (len(speed)):
            if k < 1:
                time.append(0)
            else:
                time.append(temp_time[k]+ time[k-1])
        time = np.array(time)

        battery = []
        # Energy is the relationship between power and time, power is converted
        # to kW and time is converted to hour
        for m in range(len(speed)):
            if m < 1:
                battery.append(capacity - (power[m]/1000) * (time[m]/3600))
            else:
                battery.append(battery[m-1] - (power[m]/1000) *
                               ((time[m] - time[m-1])/3600))
        battery = np.array(battery)

        return battery
```

```
#-------------------------------------------------------------------------------
def Battery_Charge(time_start, tilt, azim, Lat, Long
                   , Alt, GHI, albedo, temp, wind):
    charge = 0
    ppp = []
    for i in range (len(GHI)):
        time = datetime.datetime.strptime(time_start, '%Y-%m-%d %H:%M:%S')
               + datetime.timedelta(hours=i)
        GPOA = PV.G_POA(math.degrees(tilt), azim, time, Lat
                        , Long, Alt, GHI[i], albedo)
        # print(GPOA)
        Power = PV.Power_Out(GPOA, temp, 2, wind)
        # print(Power)
        charge += Power/1000
    return charge
```

GUI Code



```
#-------------------------------------------------------------------------------
root = Tk()

root.wm_title("2016 Nissan Leaf")
root.bind('<Escape>',lambda e: root.destroy())
root.config(background = "#F0F0F0")

Left_Frame = Frame(root, width=300, height=600)
Left_Frame.grid(row=0, column=0, sticky=N)
Label(Left_Frame, text="Options:").grid(row=0, column=0,
                                        padx=10, pady = 10, sticky=W)
Middle_Frame = Frame(root, width=300, height=600)
Middle_Frame.grid(row=0, column=1)
Right_Frame = Frame(root)
Right_Frame.grid(row=0, column=2, sticky=N)
canvas = Canvas(Left_Frame, height=150, width=250)  # A tk.DrawingArea.
canvas.grid(row=13, column=0, sticky=N)
canvas1 = Canvas(Left_Frame, height=150, width=250)  # A tk.DrawingArea.
canvas1.grid(row=15, column=0, sticky=N)
label = Label(Left_Frame, text=" ")
label.grid(row=12, column=0, sticky=NW, padx=10)
label3 = Label(Left_Frame, text=" ")
label3.grid(row=14, column=0, sticky=NW, padx=10)
label1 = Label(Left_Frame, text="")
label1.grid(row=13, column=0, sticky=SW)
label2 = Label(Left_Frame, text="")
label2.grid(row=13, column=0, sticky=SE)
label4 = Label(Left_Frame, text="")
label4.grid(row=15, column=0, sticky=SW)
label5 = Label(Left_Frame, text="")
label5.grid(row=15, column=0, sticky=SE)
Slide = Scale(Middle_Frame, orient=HORIZONTAL, length=800, showvalue=0)
Slide.grid(row=3, column=0, padx=10, pady=10)

Button_Motor = Button(Left_Frame, text="Show Synchronous Motor Characteristics",
                      command=lambda: Motor_Char(Middle_Frame, Right_Frame,
                                                 label, label1, label2, label3,
                                                 label4, label5, canvas,
                                                 canvas1, Slide))
Button_Motor.grid(row=1, column=0, padx=10, sticky=W)

Button_Forcasting = Button(Left_Frame, text="Show GHI Forecasting",
                      command=lambda:GHI.GUI_Plot(Middle_Frame, Right_Frame,
                                                  label, label1, label2,
                                                  label3, label4, label5,
                                                  canvas, canvas1, Slide))
Button_Forcasting.grid(row=2, column=0, padx=10, sticky=W)

Button_Weather = Button(Left_Frame, text="Show Current Weather Worecast",
                      command=Weather)
Button_Weather.grid(row=3, column=0, padx=10, sticky=W)

User_Power = StringVar(root)
User_Power.set(80000)
Label(Left_Frame, text="Set Power Limitation If Desired (W):").
                           grid(row=4, column=0, padx=10, sticky=W)
option = Entry(Left_Frame, textvariable=User_Power)
option.grid(row=4, column=1, padx=10, sticky=W)
```

```
User_Speed = StringVar(root)
User_Speed.set(120)
Label(Left_Frame, text="Set Speed Limitation If Desired (km/h):").
                            grid(row=5, column=0, padx=10, sticky=W)
option = Entry(Left_Frame, textvariable=User_Speed)
option.grid(row=5, column=1, padx=10, sticky=W)

test1 = IntVar()
cc=Checkbutton(Left_Frame,text="Adapt Speedprofile To Fit Battery Capacity?",
               variable=test1)
cc.grid(row=6, column=0, padx=10, sticky=W)

test3 = IntVar()
cc1=Checkbutton(Left_Frame,text="Include Contribution From Solar Panels?",
                variable=test3)
cc1.grid(row=7, column=0, padx=10, sticky=W)

test4 = IntVar()
cc1=Checkbutton(Left_Frame,text="Real Time Speed Adaption?",variable=test4)
cc1.grid(row=8, column=0, padx=10, sticky=W)

Label(Left_Frame, text="Select Plot Parameters:").grid(row=9,
                                                       column=0,
                                                       padx=10,
                                                       sticky=W)
test2 = StringVar()
test2.set('Battery Capacity vs Distance')
choices = {'Speed vs Distance', 'Power vs Distance',
           'Battery Capacity vs Distance'}
Plot_Popup = OptionMenu(Left_Frame, test2, *choices)
Plot_Popup.config(width=len(max(choices, key=len)))
Plot_Popup.grid(row=9, column=1, padx=10, sticky=W)

Button_Power = Button(Left_Frame, text="Plot",
                      command=lambda: plot_Speed_Power(User_Power.get(),
                                                       User_Speed.get(),
                                                       Middle_Frame,
                                                       Right_Frame,
                                                       label, label1,
                                                       label2, label3,
                                                       label4,
                                                       label5,canvas,
                                                       canvas1,Slide))
Button_Power.grid(row=10, column=0, padx=10, sticky=W)

Label(Left_Frame, text="Slider Values:").grid(row=11, column=0, padx=10,
                                              pady = 10, sticky=W)

#Right Fram
Label(Middle_Frame, text="Plot Area:").grid(row=0, column=0, padx=10,
                                            pady = 10, sticky=W)

canc = Canvas(Middle_Frame, width = 1000, height = 700, bg="white")
canc.grid(row=1, column=0, padx=10)


root.mainloop()
```